P. 1877/87

11-12 1987

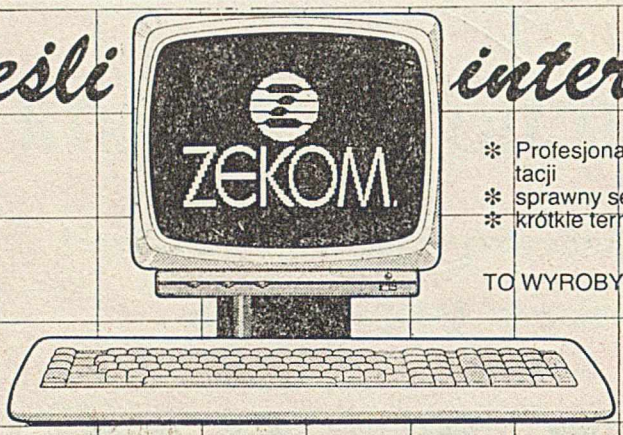
informatyka

Przetwarzanie tekstów

WYDAWNICTWO NOT SIGMA

Cena 300 zł

Jeśli interesuje Państwa



* Profesjonalny sprzęt o wysokiej jakości, niezawodny w eksploatacji
* sprawny serwis
* krótkie terminy dostaw, lub dostawy natychmiastowe

TO WYROBY ZEKOMU SĄ DO PAŃSTWA DYSPOZYCJI

Zakład Elektroniki Komputerowej

oferuje terminale ekranowe:

MV 2580 Standard VT 52 firmy DEC / odpowiednik MERA 7953. Przeznaczony do pracy w systemach komputerowych wyposażonych w kanał transmisji V 24 lub pętli prądowej 20/60 mA – jako końcówka zdalnego dostępu.

MV 2581 Odpowiednik MERA 7911 N.
Przeznaczony do pracy w systemach komputerowych ODRA 1300 wyposażonych w jednostkę sterującą MERA 7802.

MV 2582E Odpowiednik terminala typu 7181/2 firmy ICL.
Przeznaczony do pracy w systemach komputerowych ODRA 1300, ICL 1900, ICL 2900, ICL system 4.

MR 1240 Odpowiednik MERA 7951.
Przeznaczony do wprowadzania danych do systemu MERA 9150 lub systemu REDIFON.

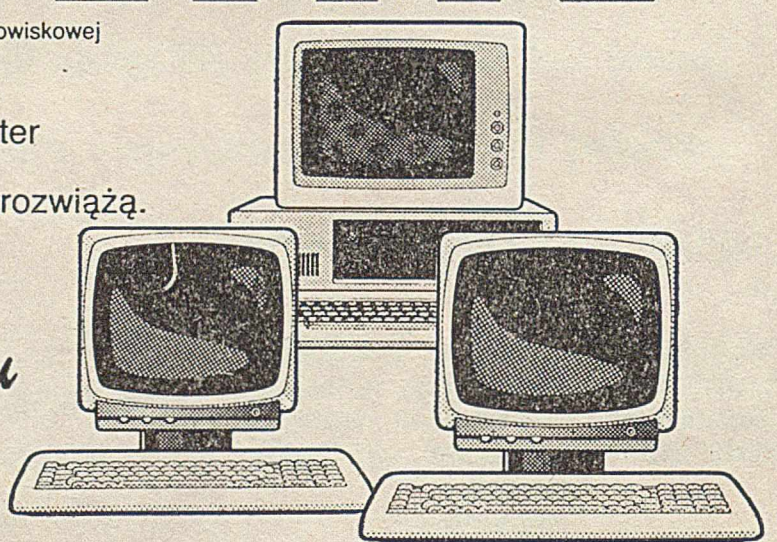
ZAKŁAD ELEKTRONIKI KOMPUTEROWEJ ul. Makowa 8, 91-480 Łódź tel. 34 30 49

multiTe™

Zestawy wieloterminalowe do pracy wielostanowiskowej
z komputerem klasy IBM PC/XT/AT

Jeśli chcecie Państwo
lepiej wykorzystać swój komputer
to zestawy multiTe
sprawnie i szybko ten problem rozwiążą.

Liczba stanowisk stosownie do potrzeb.
Możliwe nawet zestawy 8-terminalowe.



Nasze terminale gwarancją sukcesu

ZEKOM ZAKŁAD ELEKTRONIKI KOMPUTEROWEJ
Skrytka pocztowa nr 35, 90-955 Łódź 8 tel. 34-30-49

EO/307/87

P.1877/87

informatyka

11-12 1987

Nr 11-12

Miesięcznik Rok XXII

Listopad-grudzień 1987

Organ Komitetu Naukowo-Technicznego NOT ds. Informatyki

KOLEGIUM REDAKCYJNE:

Mgr Jarosław DEMINET, dr inż. Wacław ISZKOWSKI, mgr Teresa JABŁOŃSKA (sekretarz redakcji), Władysław KLEPACZ (redaktor naczelny), dr inż. Marek MACHURA, Maria PAWLAK (sekretarz redakcji), dr inż. Wiktor RZECZKOWSKI, mgr inż. Jan RYŻKO, mgr Hanna WŁODARSKA, dr inż. Janusz ZALEWSKI (zastępca redaktora naczelnego)

PRZEWODNICZĄCY RADY PROGRAMOWEJ:

Prof. dr hab. Juliusz Lech Kulikowski

Materiałów nie zamówionych redakcja nie zwraca

Redakcja: 00-517 Warszawa, ul. Mickiewicza 18 m. 17, tel. 39-14-34

Zakł. Graf. „Tamka". Zam. 0799-1300/87. Obj. 6,5 ark. druk. Nakład 8.300 egz. U-36.

ISSN 0542-9951, INDEKS 36124

Cena egzemplarza 300 zł
Prenumerata roczna 1800 zł

WYDAWNICTWO NACZELNA ORGANIZACJA TECHNICZNA

SIGMA

CZASOPISM I KSIĄŻEK TECHNICZNYCH

00-950 Warszawa
skrytka pocztowa 1004
ul. Biała 4

W NUMERZE: str.

Specyfikacje a programy (2)
Cliff B. Jones 2

Zarządzanie rozproszonymi danymi — przegląd problematyki (2)
Witold Staniszkis 7

Systemy przetwarzania tekstów
Kazimierz Bieńkowski 11

Język awk — przetwarzanie wzorców
oprac. Zdzisław Płoski 16

Maszynowe tłumaczenie tekstów
Mieczysław Kłopotek 20

Redaktor 2000 — mikrokomputerowy procesor tekstów
Tadeusz Kuroczycki 24

Chiwriter — procesor tekstu (1)
Zdzisław Płoski 27

Wordstar 3.30 — zasady działania i sposób użytkowania (3)
Stanisława Ossowska 30

Struktura systemu operacyjnego PC-DOS (3)
oprac. Anna Syfert, Andrzej Raznowiecki 32

Język C — konstruowanie, interpretowanie i wyznaczanie wartości wyrażeń
Jan Bielecki 37

Z KRAJU 41
INFOPRO '87
INFOSYSTEM '87

ZE ŚWIATA 43
Głosowy dostęp do baz danych
Systemy ekspertowe w procesach technologicznych
CeBIT '87
Kto jest kim. Jean D. Ichbiach

TERMINOLOGIA 39
Język C — propozycja polskiej terminologii (1)

Spis treści rocznika I, II, III, IV

W NAJBLIŻSZYCH NUMERACH:

- Cykl artykułów Antoniego Kreczmara o językach obiektowych
- O stanie i tendencjach rozwoju technologii programowania w kombinacie Robotron-Projekt w Dreźnie pisze Hans-Dieter Baumbach
- Andrzej Paprocki omawia ADA/SM — kompilator podzbioru Ady dla komputerów SM-4
- Jednostkę centralną Mazovii 1016 — podstawowy moduł funkcjonalnego 16-bitowego profesjonalnego komputera — opisuje Janusz Popko
- Anna Syfert i Andrzej Raznowiecki kontynuują cykl artykułów o strukturze systemu operacyjnego PC-DOS
- Piotr Zielczyński opisuje system Turbo Prolog i porównuje kompilatory i interpretery języka Prolog aktualnie dostępne na rynku
- Jan Bielecki prezentuje w dalszym ciągu język C, tym razem — wskazania, adresy i konwersje
- Robert Chyła i Krzysztof Musiał omawiają metody kompresji komunikatów

CLIFF B. JONES
Department of Computer Science
University of Manchester
Wielka Brytania

Specyfikacje a programy (2)

W tej części artykułu przedstawiono szerszy przykład ilustrujący różne aspekty podejścia systematycznego do projektowania metodą VDM. W zakończeniu omówiono różnice między VDM a innymi metodami.

ETAPY ROZWIJANIA SPECYFIKACJI

Specyfikacja pojedynczej operacji zawiera warunek wejściowy i warunek wyjściowy. Jak wspomniano wcześniej, pozwala to na definiowanie operacji częściowych i niedeterministycznych. Intuicyjnie jasne jest, że implementowana operacja może kończyć działanie dla większej liczby argumentów niż wynika to ze specyfikacji (tzn. ma większą dziedzinę, czyli jest określona dla większej liczby danych wejściowych) lub że liczba możliwych do otrzymania wyników dla dowolnych danych wejściowych jest mniejsza od liczby dopuszczalnych odpowiedzi (tzn. jest bardziej określona). Implementacje programów pisze się w pewnym języku implementacji, stąd wypływa konieczność znalezienia metody porównania znaczenia specyfikacji i programów. Dogodnym sposobem jest definiowanie ich obu jako zbioru stanów z żądaniem skończoności obliczeń oraz posłużenie się relacją semantyczną definiującą możliwe wyniki:

(S,R)

gdzie S jest podzbiorem zbioru stanów $\sum$, a R jest relacją nad $\sum$.

Para, warunek wejściowy i wyjściowy, może być przedstawiona w postaci:

$$\left(\{\sigma \in \textstyle\sum \mid_{pre(\sigma)}\},\ \{(\overline{\sigma},\sigma) \in \textstyle\sum \times \sum \mid_{post(\overline{\sigma},\sigma)}\}\right)$$

Powyższe idee semantyki denotacyjnej można zastosować do opisu elementów języka programowania przy użyciu tego samego modelu [10].

Para: zbiór i relacja umożliwia precyzyjne zdefiniowanie pojęcia **spełnialności** (sat) opisanego intuicyjnie jako:

$$(S_1, R_1)\,\mathbf{sat}\,(S_2, R_2) \Leftrightarrow S_2 \subseteq S_1 \wedge S_2 \lhd R_1 \subseteq R_2$$

O kolejnych etapach rozwijania specyfikacji należy dowieść, że są poprawne względem relacji sat.

Dwie rzeczy czynią specyfikacje dużo krótszymi od programów. Używają one obiektów nie istniejących w większości języków programowania oraz posługują się warunkami wyjściowymi, które nie określają sposobu wyznaczania wyniku. Oba te chwyty trzeba wykluczyć podczas projektowania programu. Elementy danych otrzymuje się w etapach **ukonkretniania danych**, w kolejnych etapach **dekompozycji operacji** przekształca się składniki operacji tak, aby spełniały warunki wyjściowe. Konstruowanie każdego dużego systemu przebiega w wielu krokach. Doświadczenia zdobyte w metodzie VDM sugerują, że pierwsze — wysokiego poziomu — etapy projektowania obejmują ukonkretnianie danych, a etapy późniejsze, niskiego poziomu, obejmują dekompozycje operacji. Poniższy przykład ilustruje oba rodzaje etapów konstruowania i tłumaczy pojęcie **obowiązków dowodzenia** (ang. proof obligations), zapewniających poprawność tych etapów.

Ważną rolę w tej metodzie konstruowania programów odgrywa jej **kompozycyjność** (ang. compositionality). Jej idea opiera się — najkrócej mówiąc — na tym, aby decyzja podjęta w pewnym etapie konstruowania nie była zakłócona decyzjami późniejszymi. Być może najłatwiej jest to zobaczyć na przykładzie etapów dekompozycji operacji. Operację początkową OP rozkłada się, na przykład, na dwie podoperacje OPA i OPB, podejmując decyzję o tym, że jedna zostanie wykonana po drugiej. Można opisać obie podoperacje i zweryfikować decyzję przez spełnienie odpowiednich obowiązków dowodzenia. W wyniku dalszego konstruowania można w realizacji podoperacji OPA otrzymać instrukcje pętli. W kompozycyjnej metodzie konstruowania ten etap projektu można zweryfikować bez potrzeby odnoszenia się do kontekstu, w jakim pojawiła się podoperacja OPA; wystarczy rozważyć tylko jej specyfikację. W metodach kompozycyjnych programy sekwencyjne konstruuje się bez większego trudu. Znacznie trudniejsze są programy ze wzajemnie współzależnymi procesami równoległymi. Niektóre aspekty odpowiednich prac przy użyciu metody VDM przedstawiono w [11], a bardziej aktualne badania nad logiką temporalną opisano w [2].

Do ilustrowania kolejnych etapów konstruowania programu posłuży poniższy przykład. Prosta specyfikacja zakłada istnienie zbioru obiektów pewnego danego typu X. Stan początkowy jest zbiorem pustym:

$$s_0 = \{\}$$

Przykładowa specyfikacja operacji dodania nowego elementu do zbioru ma następującą postać:

ADD (e : X)
ext wr s : set of X
pre $e \notin s$
post $s = \overline{s} \cup \{e\}$.

Operacja usuwania elementu jest następująca:

DELETE (e : X)
ext wr s : set of X
per $e \in s$
post $s = \overline{s} - \{e\}$

Operację przynależności elementu do zbioru można opisać tak:

ISPRESENT (e : X) r : B
ext rd s : set of X
post $r \Leftrightarrow e \in s$

Specyfikacja ta jest banalna, gdyż wybrane obiekty dokładnie odpowiadają problemowi. Formalnie rzecz biorąc, należy udowodnić istnienie implementacji (ang. implementability proof obligation) dla każdej operacji. Tutaj wynik ten otrzymuje się natychmiast z całkowitości operatorów na zbiorach.

Jedną z reprezentacji zbioru, uwzględniającą efektywność dostępu do elementów, jest **drzewo binarne**. Drzewa binarne:

- mają co najwyżej dwie (być może nil) gałęzie i jeden element (typu X) w każdym węźle;
- są zbudowane tak, że wszystkie elementy lewej gałęzi danego węzła są mniejsze, a wszystkie elementy prawej gałęzi są większe niż element tego węzła [1].

Formalnie strukturę danych Setrep definiuje się jako:

Setrep = [Node]
Node : : lt : Setrep
mv : X
rt : Setrep

gdzie [2]:

inv-Node(mk-Node(lt, mv, rv)) $\triangleq$
$(\forall lv \in retrns(lt) \cdot lv < mv)$
$\wedge\ (\forall rv \in retrns(rt) \cdot mv < rv)$

retrns : Setrep $\rightarrow$ set of X

retrns(sr) $\triangleq$ cases sr of

---

[1]) Dla zwięzłości relację porządku oznaczono przez „<".

[2]) Efektywność wymaga ważenia drzew (ang. balancing). Dla zwięzłości, tego żądania nie ujęto tu formalnie.

nil → { }
mk-Node(lt, mv, rt) →
retrns(lt) ∪ {mv} ∪ retrns(rt)
end

Definiowanie takiej reprezentacji jest pierwszym etapem **ukonkretniania danych** (ang. data reification). Krokiem następnym jest określenie związku między strukturą abstrakcyjną a jej reprezentacją. Dla każdego obiektu abstrakcyjnego (set of X) istnieje wiele reprezentacji (Setrep). Dlatego wygodnie będzie zdefiniować związek między abstrakcją a reprezentacją jako funkcję ze zbioru reprezentacji na zbiór obiektów abstrakcyjnych (w VDM funkcja ta nosi nazwę **funkcji odzyskania** — ang. retrieve function). W tym prostym przypadku jest to dokładnie funkcja retrns. Warto zauważyć, że jest to funkcja całkowita nad Setrep. Następnym krokiem ukonkretniania danych jest ustalenie obowiązku dowodzenia zwanego **odpowiedniością** (ang. adequacy). Żąda on, aby dla każdego stanu abstrakcyjnego istniała co najmniej jedna reprezentacja:

$\forall s \in \text{set of } X \cdot \exists sr \in \text{Setrep} \cdot \text{retrns}(sr) = s$

Powinno się to formalnie udowodnić przez indukcję względem generatorów zbioru; tu można poprzestać tylko na uzasadnieniu nieformalnym.

Po ustaleniu podstawowych właściwości reprezentacji danych należy zdefiniować każdą z operacji na zbiorze Setrep.
Przykładowo,

ADD_1 (e : X)
ext wr sr : Setrep

pre $e \in \text{retrns}(sr)$
post $\text{retrns}(sr) = \text{retrns}(\overline{sr}) \cup \{e\}$

jest specyfikacją operacji odzwierciedlającej zachowanie operacji ADD. Warunek wyjściowy jest tu niedeterministyczny, gdyż — poza oczywistymi przypadkami — akceptuje wiele możliwych wyników. Jest to przykład wykorzystania niedeterminizmu podczas projektowania. Nawet jeśli celem jest otrzymanie programu deterministycznego, to taki warunek wyjściowy pozwala podjąć decyzję o użyciu drzew binarnych bez decydowania o wyborze konkretnego algorytmu ważenia drzew.

Dla wyjaśnienia można przytoczyć bardziej rozbudowaną definicję warunku wyjściowego operacji ADD (posłużono się ideą **usyntaktycznienia operacji**), przedstawioną na wydruku 1. Użyto tu rekurencji pamiętając, że warunek wyjściowy jest tylko predykatem (w predykacie nie można wywoływać operacji).

ADD_1 (e : X)
ext wr sr : Setrep
pre e ∈ retrns(sr)
post $\overline{sr}$ = nil ∧ sr = mk-Node(nil, e, nil) ∨
$\overline{sr} \in$ Node ∧
let mk-Node(lt, mv, rt) = $\overline{sr}$ in
e < mv ∧
∃ lt ∈ Setrep · post-ADD_1(e, lt, lt)
∧ sr = mk-Node(lt, mv, rt) ∨
e > mv ∧
∃ rt ∈ Setrep · post-ADD_1(e, rt, rt)
∧ sr = mk-Node(lt, mv, rt)

Wydruk 1. Rozbudowana definicja warunku wyjściowego operacji ADD

Dowodzenie istnienia implementacji nie jest łatwe na tym poziomie projektowania. Należy dowieść, że algorytm określa obiekt spełniający niezmiennik typu danych (inv-Node). Stąd:

$\forall \overline{sr} \in \text{Setrep}, e \in X \cdot \text{pre-ADD}_1(e, \overline{sr}) \Rightarrow$
$\exists sr \in \text{Setrep} \cdot \text{post-ADD}_1(e, \overline{sr}, sr$

Formalne dowody można znaleźć w [12].

Przykład ukierunkowanej na modele specyfikacji operacji DELETE jest następujący:

DELETE_1 (e : X)
ext wr sr : Setrep

pre $e \in \text{retrns}(sr)$
post $\text{retrns}(sr) = \text{retrns}(\overline{sr}) - \{e\}$

Tutaj usunięcie niedeterminizmu jest trudniejsze. Projekt algorytmu rekurencyjnego pozostawiono Czytelnikowi jako ćwiczenie.

Model operacji ISPRESENT można wyspecyfikować jako:

ISPRESENT_1 (e : X) r : B
ext rd sr : Setrep
post $\overline{sr} = \text{nil} \wedge \text{not } r \vee$
$\overline{sr} \in \text{Node} \wedge$
let mk-Node$(\overline{lt}, mv, \overline{rt}) = \overline{sr}$ in
$e = mv \vee$
$e < mv \wedge \text{post-ISPRESENT}_1(e, lt(\overline{sr}), r) \vee$
$e > mv \wedge \text{post-ISPRESENT}_1(e, rt(\overline{sr}), r)$

Każdy z tych modeli musi spełniać bardziej abstrakcyjną specyfikację. Dla operacji ukonkretniania danych należy dowieść:

$\forall sr \in \text{Setrep} \cdot \text{pre-OP}(\text{retrns}(sr)) \Rightarrow \text{pre-OP}_1(sr)$

dla części dziedzinowej, oraz:

$\forall \overline{sr}, sr \in \text{Setrep} \cdot \text{pre-OP}(\text{retrnr}(\overline{sr})) \wedge \text{post-OP}_1(\overline{sr}, sr) \Rightarrow$
$\text{post-OP}(\text{retrns}(\overline{sr}), \text{retrns}(sr))$

dla części wynikowej. Reguły te wymagają oczywiście rozszerzenia obejmującego argumenty i wyniki.

Ze względu na wczesne występowanie etapów ukonkretniania danych w procesie projektowania, należy te dowody przeprowadzić raczej formalnie. Przykład formalnego dowodu znajduje się w rozdziale 8.3 książki [12].

Powyższe struktury drzewiaste są ilustracją zasadniczej idei drzew binarnych. Struktury te nie mają reprezentacji w większości języków programowania ze względu na ich rekurencyjną naturę. W Pascalu na przykład, trzeba w tym celu użyć wskaźników i obiektów alokowanych na stogu (ang. heap). W notacji VDM można opisać je jako:

root = [Ptr]
Heap = map Ptr to Node_2
Node_2 : : lt : [Ptr]
mv : X
rt : [Ptr]

gdzie:

inv-Heap : [Ptr] × Heap → B

inv-Heap(p, m) △ . . .

Następnym krokiem jest określenie według powyższego wzoru funkcji odzyskania:

retrsr : [Ptr] × Heap → Setrep

retrsr(p, h) △ if p = nil
then nil
else let mk-Node_2 (lt, mv, rt) = h (p) in
mk-Node retrns (lt, h), mv, retrns (rt, h))

Na tym etapie ukonkretniania, struktury danych mają postać podobną jak w potencjalnych językach implementacji. Jednak nadal takie specyfikacje nie są wykonywalne. Warunki wejsciowe i wyjściowe należy rozłożyć na ciągu wyrażeń. W metodzie VDM etap ten nosi nazwę **dekompozycji operacji** (ang. operation decomposition). Przytoczony tu przykład rozwinięto w książce [12]. Tutaj niedostatek miejsca każe posłużyć się krótszymi przykładami.

Proces dekompozycji operacji również pociąga za sobą obowiązek dowodzenia. Powstaje pytanie, skąd bierze się ten obowiązek i jak zawrzeć w programach uzasadnienie ich poprawności. Idee te i ich zastosowanie w konstruowaniu programów omówiono w artykule [13].

Poniżej zaprezentowano reguły dowodzenia dla kilku prostych konstrukcji języków programowania. Ogólnie, postać tych reguł jest podobna do postaci w logice. Reguły te można stosować, o ile podoperacje spełniają pewne właściwości i w konkluzji kombinacja tych podoperacji spełnia pewne (inne) właściwości. Dzięki tym regułom łatwiej udowodnić, że fragmenty programu spełniają podobne specyfikacje. Można więc wykazać, że:

if i < 0 then i,j := —i, —j else skip

spełnia specyfikację:

MAKEPOS ( )
ext wr i : Z,
wr j : Z
pre true

post $0 \leqslant i \wedge i*j = \bar{i}*\bar{j}$

W małych przykładach wygodnie jest specyfikacje i implementacje zapisywać razem:

MAKEPOS
ext wr i : Z, wr j : Z
pre true

if i < then i,j := —i,—j else skip

post $0 \leqslant i \wedge i*j = \bar{i}*\bar{j}$

Program zapisuje się tak wówczas, gdy wcześniej udowodniono, że kod spełnia specyfikację. Można czasem pominąć nazwę operacji i jej parametry zewnętrzne, jeśli jasno wynikają z kontekstu.

Naturalnym rozszerzeniem tego stylu jest specyfikowanie podoperacji zamiast ich kodu (wydruk 2). Podany przykład należy rozumieć tak, że złożenie dwóch podoperacji MAKEPOS i POSMULT będzie spełniać specyfikację operacji MULT. Podoperacje te nie są jeszcze zakodowane — podane są tylko ich specyfikacje [3]).

```
MULT
ext wr i, j, m : Z
pre true
        MAKEPOS
        ext wr i, j : Z
        pre true
        post i ≥ 0 ∧ i * j = ī * j̄
        ;
        POSMULT
        pre i ≥ 0
        post m = ī * j̄
post m = ī * j̄
```

Wydruk 2. Przykład zastąpienia kodu specyfikacją

Reguły dowodzenia są nieco podobne do reguł dowodzenia w logice. Mówiąc nieściśle, dla każdej konstrukcji językowej istnieje oddzielna reguła dowodzenia. Aby przedstawić reguły dowodzenia w zwartej postaci, warunki wejściowe i wyjściowe umieszczono (w nawiasach) przed i po odpowiadających im segmentach kodu:

{wejściowy} S {wyjściowy}

Czasami warunek wyjściowy musi zawierać tę samą informację, którą zawiera warunek wejściowy. Niech P z kreską oznacza zdanie logiczne P, w którym wszystkie zmienne wolne występują z haczykiem [4]). Wówczas stosowana reguła dowodzenia ma postać:

$$\frac{\{P\}\,S\,\{R\}}{\{P\}\,S\,\{\bar{P} \wedge R\}}$$

Przykładowo, z faktu

$\{fn = 1\}\,FACTB\,\{fn = \bar{fn} * \bar{n}!\}$

wynika:

$\{fn = 1\}\,FACTB\,\{fn = \bar{fn} * \bar{n}! \wedge \bar{fn} = 1\}$

i stąd:

$\{fn = 1\}\,FACTB\,\{\bar{fn} = \bar{n}!\}$.

[3]) Projekt można przedstawić jako złożenie (opisanych) podproblemów. Metoda kompozycyjna konstruowania programu umożliwia zweryfikowanie projektu przy użyciu specyfikacji jego (syntaktycznych) podprogramów. Pojedynczy krok konstruowania jest więc niezależny od dalszych kroków w tym sensie, że każdej implementacji podprogramu można użyć do zaimplementowania całej specyfikacji. W metodzie niekompozycyjnej poprawność kroku konstruowania może zależeć od dalszej rozbudowy podprogramów. Nie jest zbyt trudno znaleźć metody kompozycyjne dla programów sekwencyjnych. Dla programów z oddziaływaniem procesów równoległych zadanie jest znacznie trudniejsze [2, 11].

[4]) W tekście polskojęzycznym są to zmienne z kreską — przyp. red.

Warto zauważyć, że oznaczenie kreską zdania P, a stąd i jego zmiennych wolnych, ma kluczowe znaczenie — nie jest prawdą, że ostatecznie:

$fn = 1$

Najprostszą kombinacją dwóch operacji jest ich sekwencyjne wykonanie. Jest zrozumiałe, że stan końcowy pierwszej operacji musi spełniać warunek wejściowy drugiej. Aby to zapisać, trzeba rozróżnić właściwości relacyjne i właściwości pojedynczych stanów, otrzymane po wykonaniu pierwszej operacji. Rozróżnienie to ułatwiają odpowiednie nazwy funkcji o wartościach prawda lub fałsz:

$P: \sum \to B$

$R: \sum \times \sum \to B$

Niech

$R_1 \mid R_2$

oznacza relację:

$\exists \sigma_i \cdot R_1(\bar{\sigma}, \sigma_i) \wedge R_2(\sigma_i, \sigma)$

Wówczas reguła sekwencji ma postać:

$$\frac{\{P_1\}\,S_1\,\{P_2 \wedge R_1\},\ \{P_2\}\,S_2\,\{R_2\}}{\{P_1\}\,S_1; S_2\,\{R_1 \mid R_2\}}$$

Predykat P_2 można uważać za dokonany przez projektanta wybór sprzężenia (ang. interface), między S_1 a S_2, natomiast R_1 i R_2 ustalają funkcjonowanie obu składowych.

Pierwszy czynnik koniunkcji warunku wyjściowego operacji MAKEPOS (wydruk 2) zawiera się w warunku wejściowym operacji POSMULT. Definiuje on sprzężenie między tymi operacjami (przed ich zakodowaniem). Warunek wyjściowy POSMULT jest taki sam, jak warunek wyjściowy MULT, lecz położenie tego pierwszego wskazuje, że jego zmienne oznaczone haczykiem (lub kreskami — przyp. red.) odnoszą się do wartości otrzymanych między wykonaniem operacji MAKEPOS i POSMULT. Drugi czynnik warunku wyjściowkego MAKEPOS jest najprostszym wyrażeniem (dla R_1 w regule sekwencji), zapewniającym spełnienie warunku wyjściowego operacji MULT. Warunek wyjściowy MAKEPOS można bardziej szczegółowo przedstawić jako:

$\text{post-MAKEPOS}(\bar{i}, \bar{j}, \bar{m}, i, j, m) \triangleq i*j = \bar{i}*\bar{j} \wedge m = \bar{m}$

Brak m w zewnętrznej klauzuli operacji MAKEPOS wynika z drugiego czynnika koniunkcji.
Podobnie:

$\text{post-POSMULT}(\bar{i}, j, \bar{m}, i, j, m) \triangleq m = \bar{i}*j$

skąd:

pots-MAKEPOS | post-POSMULT

czyli:

$\exists i_i, j_i, m_i \cdot i_i * j_i = \bar{i}*\bar{j} \wedge m_i = \bar{m} \wedge m = i_i * j_i$

i stąd otrzymujemy:

$m = \bar{i}*\bar{j}$

W praktyce na ogół nie przeprowadza się tych dowodów w sposób tak formalny. Łatwo sprawdza się tekst zapisany tak, jak dla operacji MULT. Należy jedynie zwrócić uwagę na związek zmiennych opatrzonych haczykiem (lub kreską — przyp. red.) z ich wartościami początkowymi dla odpowiedniej operacji. Czytanie tekstu jest ułatwione przez jego układ graficzny (zagnieżdżanie). Uogólnienie reguły na ciąg więcej niż dwóch zdań nie nastręcza żadnych trudności. Reguła dowodzenia dla zdań warunkowych ma postać:

$$\frac{\{P \wedge B\}\,TH\,\{R\},\ \{P \wedge \text{not } B\}\,EL\,\{R\}}{\{P\}\ \text{if B then TH else EL}\{R\}}$$

Postać tej reguły sugeruje, że projektant ma tu mniejszą swobodę niż przy wyborze przypadków. Warto zobaczyć jak jest w istocie, sięgając choćby po tak prosty przykład jak mnożenie:

MAKEPOS
ext wr i, j ; Z
pre true
if i < 0
then pre i < 0

post $0 \leqslant i \wedge i = -\bar{i} \wedge j = -\bar{j}$

else pre $0 \leqslant i$

post $0 \leqslant i \wedge i = \bar{i} \wedge j = \bar{j}$

post $0 \leqslant i \wedge i*j = \bar{i}*\bar{j}$

Formalnie w uzasadnieniu korzysta się z reguły pozwalającej na użycie mocniejszego warunku wejściowego lub słabszego warunku wyjściowego:

$$\frac{PP \Rightarrow P, \{P\}S\{RR\}, RR \Rightarrow R}{\{PP\}S\{R\}}$$

Warunek wyjściowy obu zdań z przykładu ma wyrażać intencje projektanta, a nie kopiować warunek wyjściowy operacji MAKEPOS. W ten sposób — przy większym projekcie — projektuje się składowe w oderwaniu od ich kontekstu.

Reguła dowodzenia dla iteracji jest — jak należało się spodziewać — najbardziej interesująca. Jej ogólna postać jest następująca:

$$\frac{\{P \wedge B\}S\{P \wedge R\}}{\{P\}\ \text{while } B \text{ do } S\{P \wedge \text{not } B \wedge R^*\}}$$

Relacja R musi być dobrze ufundowana (ang. well founded) i przechodnia. Wyrażenie logiczne R nie jest zwrotne, np.

$x < \bar{x}$

Ponieważ ciało pętli może nie wykonać się ani razu, pętla while może wcale nie zmienić stanu. Stąd w ogólnej postaci warunku wyjściowego wystarczy przyjąć R* (przechodnie domknięcie R), np.:

$x \leqslant \bar{x}$

Założenie, że R jest dobrze ufundowane (skąd wynika, że nie jest zwrotne), jest istotne, gdyż zapewnia zakończenie wykonania pętli. Zrozumienie całej reguły nie sprawia kłopotu. Wyrażenie P jest niezmiennikiem pętli, tzn. jest prawdziwe po każdej liczbie (począwszy od zera) wykonań ciała pętli. Jest to w istocie szczególny przypadek niezmienniczego typu danych. Warto zauważyć, że niezmiennik może nie być spełniony wewnątrz ciała pętli. Fałszywość B za pętlą wynika natychmiast z istoty konstrukcji pętli. Ponownie wracając do przykładu mnożenia, można zaimplementować operację POSMULT tak, jak na wydruku 3. Czytelnik powinien uważnie sprawdzić, jak mają się termy poszczególnych wyrażeń logicznych do reguł dowodzenia. Zauważmy, że rel jest dobrze ufundowana, gdyż i jest zawsze dodatnie i nie może być w nieskończoność pomniejszane (por. inv).

```
POSMULT
ext wr i, m  Z, rd j  Z
pre 0 ≤ i
    m := 0
    ;
    pre 0 ≤ i
        while i ≠ 0 do
        inv 0 ≤ i
            i := i - 1;
            m := m + j;
        rel m = m̄ + (i - 1) * j̄ ∧ i < ī
    post m = m̄ + ī * j̄
post m = ī * j̄
```

Wydruk 3. Rozwinięcie przykładu z mnożeniem

Implementacja z wydruku 3 jest zbyt wolna, ponieważ jest liniowa względem wartości i. Na wydruku 4 przedstawiono algorytm, którego czas wykonania jest proporcjonalny do $\log_2 i$. Wykorzystuje on możliwości komputera binarnego: rozróżnianie liczb parzystych i nieparzystych (przez sprawdzanie najmniej znaczącego bitu) oraz mnożenie lub dzielenie przez 2 (przez przesunięcie reprezentacji dwójkowej liczby). Zewnętrzne pętle obu tych algorytmów są identyczne. Jednak, klauzula ext operacji POSMULT została zmieniona tak, aby umożliwić konieczne przypisanie zmiennej j. Podobnie zmieniono klauzulę rel pętli, aby zaspokajała potrzeby w ogólniejszym przypadku [5]). W obu przypadkach relacja jest przechodnia.

Rodzi się tu uzasadnione pytanie, jak utworzyć wyrażenia inv i rel? Wyprowadzanie dowodów z kodu nie stanowi głównego tematu artykułu, nie ma tu więc rozważań na ten temat. Warto jednak zauważyć, że w pewnym sensie etap dowodzenia jest czynnością odwrotną do projektowania programu. Analogicznie, w rachunku różniczkowym sprawdza się poprawność wyprowadzonych wyrażeń całkowych przez ich różniczkowanie. W [13] wykazano, jak obowiązki dowodzenia dla konstrukcji programowych wpływają na proces projektowania programu.

```
POSMULT
ext wr i, j, m : Z
pre 0 ≤ i
    m := 0
    ;
    pre 0 ≤ i
        while i ≠ 0 do
        inv 0 ≤ i
            ext wr i, j : Z
            pre i ≠ 0
                while is-even(i) do
                inv 1 ≤ i
                    i := i / 2
                    j := j * 2
                rel i * j = ī * j̄ ∧ i < ī
            post i * j = ī * j̄ ∧ i ≤ ī
            ;
            m := m + j;
            i := i - 1
        rel m + i * j = m̄ + ī * j̄ ∧ i ≤ ī
    post m = m̄ + ī * j̄
post m = ī * j̄
```

Wydruk 4. Alternatywne rozwinięcie przykładu z mnożeniem

Wiele interesujących przykładów takich dowodów dostarczają programy przeszukujące i sortujące [15]. W przypadku przeszukiwania rozpatrywany wektor pozostaje niezmieniony i dowody posługujące się relacjami wejścia-wyjścia niewiele różnią się od tych, w których stan końcowy jest opisany tylko warunkiem wyjściowym. Użyteczność ogólniejszych warunków wyjściowych staje się oczywista w przykładach sortowania, gdzie podstawowy wektor zmienia się. W następującej specyfikacji:

SORT ( )
ext wr l : seq of N
post is-ord (l) ∧ is-perm (l,l̄)

Funkcja porządkująca

is-ord : seq of N → B

i permutująca

is-perm: seq of N x seq of N → B

są oczywiste (choć interesujące może być zdefiniowanie drugiej z nich przez jej właściwości, a nie bezpośrednio).

Bardzo prostą strategią sortującą jest umieszczanie — w każdym obiegu pętli — „kolejnego" elementu na poprawnym miejscu. Tę decyzję projektową zilustrowano na wydruku 5.

```
SORT
var i : ℕ
i := 1;
while i ≠ n do
inv is-ord(l(1,...,i))
        i := i + 1
        ;
        BODY(i)
        pre i ∈ dom l
        post l(i + 1,...) = l̄(i + 1,...) ∧
                ∃ j ∈ (1,...,i) .
                    l(j) = l̄(i) ∧
                    l̄(1,...,i - 1) = del(l(1,...,i),j)
rel is-perm(l,l̄)
post is-ord(l) ∧ is-perm(l,l̄)
```

Wydruk 5. Pierwsza strategia sortowania

Równie prosty (i równie nieefektywny) algorytm sortowania polega na wyborze w każdym kroku iteracji najmniejszego z dotychczas nie ułożonych elementów i umie-

[5]) Bardzo interesujące jest porównanie tych dwóch predykatów. Pierwszy wskazuje wprost pracę, którą należy wykonać, a drugi — wyrażenie, którego wartość ma być stała.

szczeniu go na kolejnej pozycji. Ma on postać przedstawioną na wydruku 6.

W zestawieniu warunków wyjściowych podoperacji BODY (w obu przypadkach) zawiera się kwintesencja zamierzonego projektu. Skonstruowanie bardziej efektywnych algorytmów pozostawiamy zainteresowanemu Czytelnikowi.

```
SORT
var i : ℕ
i := 0;
while i ≠ n - 1 do
inv is-ord(l(1,...,i)) ∧
        ∀ m ∈ {1,...,i}, n ∈ {i+1,...} . l(m) ≤ l(n)
    i := i + 1
    ;
    BODY(i)
    pre i ∈ dom l
    post l(1,...,i-1) = l~(1,...,i-1) ∧
            ∃ j ∈ {i,...} .
                l(i) = l~(j) ∧ l(i+1,...) = del(l~(i,...), j)
rel is-perm(l, l~)
post is-ord(l) ∧ is-perm(l, l~)
```

Wydruk 6. Druga strategia sortowania

Podane wyżej reguły dowodzenia wystarczają dla warunków wyjściowych dwóch stanów (relacje wejścia-wyjścia), a są tylko trochę bardziej skomplikowane od podanych w [7]. Jak wykazano w przykładach, oddzielenie niezmienników i relacji może wspomóc proces projektowania (chociaż reguły podane w [9] są trudne, warto przekonać się o ich użyteczności). W [18] warunki wejściowe są również funkcjami logicznymi dwóch stanów.

W zależności od etapu konstruowania, projekt zapisuje się na przemian w języku programowania i w języku specyfikacji. Jednakże zadania specyfikacji i implementacji są różne.

* * *

Warto jeszcze krótko omówić inne metody specyfikacji na tle podejścia prezentowanego w tym artykule. Użyta powyżej reguła ukonkretniania danych nie jest zupełna w tym sensie, że pewnych zagadnień, użytych podczas konstruowania reprezentacji, nie da się zweryfikować. Inną, alternatywną regułę opisano w [5, 16], a ogólną motywację przedstawiono w artykule [13]. Jest ona **zupełna** w znaczeniu sprecyzowanym w podanych pracach źródłowych.

Logika, którą — począwszy od [1] — posługuje się autor tego artykułu, pozwala na użycie funkcji częściowych. Zasadniczo dowodzenie przez naturalną dedukcję wprowadzono dla symetrycznych trójwartościowych tablic logicznych Łukasiewicza w [14]. Artykuł [3] odpowie na pytanie, w jaki sposób rozwiązać problem nieokreślonych termów w wyrażeniach logicznych. W artykule [13] nawiązano do:

— użycia „równości egzystencjalnej" i problemu jej negacji;

— użycia „mocnej równości" i oddzielenia możliwych do przeprowadzenia (ang. definedness) dowodów;

— podejść, których autorami są Abrial, Blikle i Owe.

Istnieje oczywista potrzeba dalszych eksperymentów w tej dziedzinie, aby ustalić dogodne zasady dowodzenia.

Wokół związku między specyfikacjami a językami programowania rodzi się wiele ważnych pytań. Na przykład, podczas dekompozycji operacji w metodzie VDM naturalne jest jednoczesne stosowanie konstrukcji języka programowania i specyfikacji operacji. Michel Sintzoff [17] argumentuje, że specyfikacje i języki programowania stanowią część jednego continuum. Przyjmując ten punkt widzenia, nie należy stosować języka programowania jako języka specyfikacji, gdyż — w najlepszym wypadku — powoduje to utratę przejrzystości w wyrażaniu częściowości lub niedeterminizmu operacji. Co gorsze, brak oczywistych właściwości algebraicznych, takich jak łączność, utrudnia wówczas wnioskowanie na podstawie takiego tekstu. Ominięcie tych trudności jest główną zaletą języka specyfikacji. Z doświadczeń w tej dziedzinie wynika, że studenci zaznajomieni z takim językiem mają tendencję do nadużywania cech domyślnych.

Bardziej interesującym podejściem jest użycie części języka programowania jako podzbioru języka specyfikacji. Sposobem przeniesienia ich na wspólny grunt semantyczny jest definiowanie konstrukcji programistycznych przez tłumaczenie ich na język predykatów. Instrukcje przypisania można wówczas zastosować do rozwiązania „problemu budowy" (ang. „frame problem"). Można również — według uznania — posługiwać się predykatami. W książkach [6, 8] opisano podejścia oparte na szczególnie eleganckich regułach dekompozycji. Jednakże żadnej z nich nie można zastosować do wyprowadzania programów w przyjętym przez nas porządku, ani też do wyjaśnienia tak wielu aspektów, włączając problem zakończenia obliczeń. Warto też dodać, że bardzo interesującą metodę rozwija aktualnie Jean-Raymond Abrial. Tłumaczenie na predykaty zapewnia podstawy semantyczne, ale wydaje się gubić związek z decyzjami projektowymi. Może to prowadzić do analogicznych kłopotów jak te, które pojawiły się (por. [4]) podczas eksploatacji „generatora warunków weryfikacyjnych" (ang. „verification condition generators"). Jedną z najbardziej interesujących części książek [6, 8] jest rozszerzenie języków o właściwości algebraiczne. Łączy ono podejścia, specyfikacja-projekt-weryfikacja oraz transformacja specyfikacji, omawiane na początku artykułu. Być może należy zbadać rozwiązanie alternatywne, w którym ograniczony podzbiór predykatów będzie można tłumaczyć na programy.

Autor chciałby wyrazić podziękowania organizatorom III Jesiennej Szkoły Polskiego Towarzystwa Informatycznego, Mrągowo'86, za zaproszenie do wygłoszenia wykładu i za pozwolenie na dalsze publikowanie tego materiału. Artykuł przepisała — pod wielką presją czasu — p. Julie Hibbs. Ponadto autor dziękuje Radzie SERC (ang. Science and Engineering Research Council) za pomoc finansową.

Tłumaczył i opracował:
WOJCIECH PACHOCKI

LITERATURA

[1] Barringer H., Cheng J. H., Jones C. B.: A Logic Covering Undefinedness in Program Proofs. Acta Informatica, Vol. 21, No. 3, pp. 251—269, 1984.
[2] Barringer H., Kuiper R., Pnueli A.: Now You May Compose Temporal Logic Specifications. Proc. 16th ACM Symposium on the Theory of Computing, Washington (DC), 1984
[3] Cheng J. H., Jones C. B.: On the Haldling of Partial Functions (w przygotowaniu)
[4] Craigen D.: A Technical Review of Four Verification Systems — Gypsy, Affirm, FDM and Revised Special. August 1985
[5] He J., Hoare C. A. R., Sanders J. W.: Data Refinement Refined — Resume. Proc. ESOP'86, Robinet B., Wilhelm R. (eds.), Lecture Notes in Computer Science, Vol. 213, Springer-Verlag, 1986
[6] Hehner E. C. R.: The Logic of Programming. Prentice-Hall International, 1984
[7] Hoare C. A. R.: An Axiomatic Basis for Computer Programming. Comm. of the ACM, Vol. 12, No. 10, pp. 576—580, October 1969
[8] Hoare C. A. R. et al.: Laws of Programming — A Tutorial Paper. Technical Monograph PRG-45, Oxford University, May 1985
[9] Jones C. B.: Konstruowanie oprogramowania metodą systematyczną. WNT, Warszawa, 1984
[10] Jones C. B.: Development Methods for Computer Programs Including a Notion of Interference. Technical Monographs PGR-25, Oxford University, June 1981
[11] Jones C. B.: Specification and Design of Parallel Programs. Proc. of IFIP'83, North-Holland, 1983
[12] Jones C. B.: Systematic Software Development using VDM. Prentice-Hall International, 1986
[13] Jones C. B.: Program Specification and Verification in VDM. Marktoberdorf Summer School, 1986 (w druku)
[14] Kleene S. C.: Introduction to Metamathematics. North-Holland, 1967
[15] Knuth D. E.: Sorting and Searching. The Art of Computer Programming, Vol. III, Addison-Wesley, 1975
[16] Nipkow T.: Non-Deterministic Data Types — Models and Implementations. Acta Informatica, Vol. 22, pp. 629—661, 1986
[17] Sintzoff M.: Expressing Program Design in a Design Calculus. Marktoberdorf Summer School, 1986 (w druku)
[18] Tarlecki A.: A Language of Specified Programs. Science of Computer Programming, Vol 5, No. 1, pp. 59—81, 1985.

WITOLD STANISZKIS
Centrum Projektowania i Zastosowań Informatyki
Warszawa

Zarządzanie rozproszonymi danymi — przegląd problematyki (2)

W tej części artykułu skoncentrowano się głównie na problematyce systemów z wieloma bazami danych (multibazami).

SYSTEMY Z WIELOMA BAZAMI DANYCH

Zwykle programy aplikacyjne, działające w systemie z wieloma bazami danych, można scharakteryzować za pomocą następującej architektury abstrakcji danych:

Schemat globalny jest centralnym składnikiem systemu z wieloma bazami danych. Zawiera wszystkie elementy logicznej struktury danych wynikające z procesu abstrahowania i integracji przeprowadzonego na schematach lokalnych baz danych. Schemat ten daje zintegrowany pogląd na rozproszoną baze danych, określający fragmentację i przezroczystość miejsca oraz niezależny od kwestii związanych z heterogenicznością składników systemu.

Schematy użytkowników odpowiadają w architekturze ANSI/SPARC pojęciu schematu zewnętrznego. Schematy użytkowników zawierają wszystkie perspektywy (ang. views) oraz zmaterializowane perspektywy (ang. snapshots) zdefiniowane przez użytkowników.

Schematy lokalne, definiowane w języku opisu danych miejscowego SZBD, reprezentują strukturę logiczną odpowiadających im lokalnych baz danych, które mają być zintegrowane w systemie z wieloma bazami danych.

Schemat pomocniczy, ewentualnie również poddany fragmentacji, jest przechowywany we wszystkich węzłach sieci komputerowej. Zawiera schematy odwzorowań oraz informacje (ilościowe i inne), używane w rozwiązywaniu problemów, wynikających z niezgodności między lokalnymi bazami danych a odpowiednimi miejscowymi systemami zarządzania bazami danych.

W większości systemów, schemat globalny jest replikowany we wszystkich stanowiskach sieci komputerowej. Fakt ten oraz potrzeba uzyskiwania dostępu do danych przechowywanych w oddzielnych, geograficznie rozproszonych bazach danych tworzą nowe problemy w dziedzinie zarządzania i projektowania baz danych.

Wśród niżej przedstawionych systemów jest kilka opracowanych jako prototypy badawcze. W niektórych wypadkach, po pomyślnym zaimplementowaniu prototypów, podjęto prace nad produktami programowymi. Do tej pory nie istnieje jednak żaden produkt handlowy, w którym byłyby zadowalająco zrealizowane cechy oczekiwane w systemach z wieloma bazami danych:

● A Distibuted Database System (ADDS)

ADDS został opracowany w Amoco Production Company (USA). Jego głównym celem było połączenie baz danych dotyczących badań ropy naftowej, zainstalowanych w sieci komputerowej opartej na komputerach IBM [2]. W systemie tym schemat globalny jest relacyjny. ADDS umożliwia zintegrowany dostęp do baz danych pod nadzorem systemów Inquire, IMS/VS i SQL/DS. Udogodnienie w postaci wspólnego języka zapewnia przezroczystość dystrybucji i lokalnych baz danych. Pytania globalne są odwzorowywane na zbiór pytań lokalnych, wyrażonych w języku zapytań miejscowego SZBD. Niezgodności z lokalnym językiem zapytań są rozstrzygane po przetworzeniu pytania. Kryteriami optymalizacji są: koszt komunikacji i koszt przetwarzania logicznego. Rozproszone przetwarzanie transakcji nie jest możliwe.

● MERMAID

MERMAID jest systemem do testowania. Został opracowany w System Development Corporation (USA) z zadaniem zintegrowania relacyjnych baz danych eksploatowanych w lokalnej sieci komputerowej, zbudowanej z użyciem komputerów VAX (VAX/X i VAX/VMS) [23]. W systemie tym są dostępne bazy danych działające pod nadzorem maszyny bazy danych IDM firmy Britton Lee oraz systemów INGRES i ORACLE. Schemat globalny jest relacyjny i zawiera także wszystkie informacje pomocnicze. Pojedynczy translator akceptuje wiele spośród relacyjnych języków danych, w szczególności SQL, QUEL i ARIEL. Przetwarzanie pytań odbywa się z użyciem języka pośredniego DIL (ang. distributed intermediate language), zaprojektowanego w celu zapewnienia realizacji rozproszonych pytań. Kryterium optymalizacji pytań jest koszt komunikacji (stosowany jest rozszerzony algorytm SDD-1). Optymalizator pytań jawnie uwzględnia fragmentację i replikowanie danych. Przetwarzanie transakcji nie jest w tym systemie możliwe.

● MULTIBASE

MULTIBASE został opracowany w Computer Corporation of America w celu wyszukiwania danych z heterogenicznych rozproszonych baz danych [14, 20]. Schemat globalny zakłada funkcjonalny model danych; do formułowania pytań stosuje się język danych Daplex. Pytanie globalne jest transformowane na zbiór pytań lokalnych, które mogą być wykonywane przez lokalne SZBD. Rozstrzyganie potencjalnych niezgodności w stosunku do lokalnych języków pytań następuje na końcowym etapie przetwarzania. Zarządzanie fragmentacja nie jest bezpośrednie, natomiast jest możliwe użycie mechanizmu tworzenia struktur pochodnych w celu generowania danych globalnych na podstawie danych lokalnych. Jeżeli więc pewne lokalne jednostki danych, będące porcjami tej samej jednostki danych, są autonomicznie przechowywane w kilku lokalnych bazach danych, to schemat globalny obejmuje tylko tę globalna jednostkę danych, tj. nie zawiera żadnych odwołań do jej komponentów. W tym sensie przezroczystość fragmentacji jest w MULTIBASE zapewniona.

Optymalizacja pytań jest podzielona na dwa etapy: optymalizacje globalna i lokalna. Algorytm globalnej optymalizacji uwzględnia koszt komunikacji oraz próbuje wykorzystać możliwości przetwarzania równoległego, a algorytm optymalizacji lokalnej działa w kierunku minimalizacji czasu wyszukiwania danych z lokalnego SZBD. MULTIBASE nie zapewnia przetwarzania transakcji.

● Network Data Management System (NOMS)

NOMS został opracowany w Consorzio per la Ricerca e le Applicazioni di Informatica (Włochy) w celu wyszukiwania i rozproszonego przetwarzania danych rozmieszczonych w oddzielnych, wcześniej istniejących bazach danych oraz w plikach danych zainstalowanych w heterogenicznych sieciach komputerowych [21]. Pomyślna realizacja prototypu działającego w sieci komputerów IBM oraz VAX pod nadzorem systemów zarządzania bazami danych ADABAS, IDMS, DB/DC, RODAN i INGRES doprowadziła do opracowania systemu DQS (ang. distributed query system). Aktualnie DQS jest już systemem handlowym, działającym w sieci SNA komputerów IBM. Schemat globalny, replikowany we wszystkich węzłach sieci kompu-

terowej, jest relacyjny. Językiem użytkownika jest SQL. Schematy użytkowników są realizowane przy użyciu mechanizmu definicji perspektyw języka SQL, przy czym mogą być określone zarówno perspektywy stałe, jak i chwilowe.

Schemat pomocniczy zawiera definicje odwzorowań sterujące materializacją relacji schematu globalnego. Przetwarzanie pytań jest oparte na pośrednim języku manipulacji danymi, zaprojektowanym w celu wykorzystania inherentnych możliwości realizacji równoległej. Scentralizowana procedura planująca generuje plan dla pytania (wyrażony w języku pośrednim) na podstawie wyników optymalizacji pytania. Dodatkowy końcowy etap przetwarzania nie jest konieczny, ponieważ wszystkie dostępy do lokalnych baz danych odbywają się za pośrednictwem modułów dostępu SZBD, opracowanych przy użyciu poszczególnych języków manipulacji danymi. Przetwarzanie transakcji w systemie NOMS jest możliwe, przy czym niezbędna synchronizacja jest oparta na protokole dwufazowego kończenia (ang. two-phase commit) oraz udogodnieniach zapewnianych przez miejscowy SZBD.

● SIRIUS-DELTA

Częścią projektu SIRIUS, zainicjowanego w 1977 r. w INRIA, było opracowanie architektur systemów o wielu bazach danych (tj. z multibazą) i ich eksperymentalne zbadanie. Uznano, że w środowisku o wielu bazach danych przezroczystość miejsca nie jest konieczna [15]. Do wykonywania operacji w systemie z wieloma bazami danych opracowano język manipulacji Malpha, oparty na języku asercji Alpha, zaprojektowanym przez Codda. Język definicji danych opisuje strukturę w zależności między poszczególnymi bazami danych. Są to: zależności semantyczne, więzy integralności oraz zależności dotyczące prywatności danych. Przyjęto, że kolekcja relacyjnych baz danych jest relacyjną multibazą danych. Język Malpha umożliwia także dokonywanie aktualizacji multibazy danych.

ROZPROSZONE PRZETWARZANIE PYTAŃ

W procesie przetwarzania pytań, w systemie o rozproszonych współdzielonych danych mogą wystąpić następujące sekwencje kroków:

Translacja pytania. Źródłowa specyfikacja pytania jest transformowana do postaci drzewa pytania. Podczas tzw. etapu modyfikacji, pytania odnoszące się do perspektyw są rozszerzane do postaci pytań złożonych, uwzględniających perspektywy. Liśćmi drzewa pytania są relacje schematu globalnego. Relacje te odpowiadają relacjom bazowym scentralizowanego SZBD.

Algebraiczna transformacja drzewa pytania. Proces ten ma na celu zredukowanie rozmiaru rezultatów pośrednich pytania (zwykle reprezentowanych przez węzły w drzewie pytania) metodą przesuwania operacji selekcji i projekcji w dół drzewa pytania, możliwie jak najbliżej poziomu węzłów relacji globalnych. Identyfikowane są także wspólne podwyrażenia, dzięki czemu upraszcza się drzewo pytania.

Dekompozycja pytania globalnego. Można tu zastosować dwa podejścia: jedno polega na transformacji pytania odnoszącego się do relacji globalnej na pytania do fragmentów, drugie natomiast — na wyrażeniu pytania globalnego w postaci programu pytania w języku pośrednim, a następnie zdekomponowaniu tego programu na podprogramy pytania. Ponieważ można uzyskać wiele różnych dekompozycji, w celu wybrania „najlepszej" z nich stosuje się specjalne algorytmy optymalizacyjne.

Przetwarzanie lokalnych podpytań obejmuje fizyczny dostęp do lokalnych baz danych oraz wykonywanie operacji algebry relacyjnej na odpowiednich fragmentach. Mogą być także uzyskiwane i przetwarzane wyniki częściowe z innych węzłów systemu. Systemy heterogenicznych multibaz muszą ponadto wykonywać operacje odwzorowujące, w celu uzyskiwania relacji globalnych (lub ich fragmentów) z odpowiednich lokalnych baz danych. Strategia przetwarzania lokalnych podpytań może być zdeterminowana przez algorytmy optymalizacji pytań lokalnych.

Prezentacja rezultatów pytań. Rezultaty pytań mogą być wyświetlane bezpośrednio na terminalu użytkownika albo zapamiętane w postaci zmaterializowanych perspektyw, chwilowych lub stałych. W celu zwiększenia czytelności informacji wynikowej stosuje się m.in. formatowanie raportów lub sortowanie. Dostępne są sprzężenia (ang. interfaces) do systemów programowych: graficznych, arkuszy rachunkowych (ang. spreadsheet) oraz statystycznych.

Pierwsze dwa kroki przetwarzania pytań stosują się zarówno do systemów rozproszonych, jak i scentralizowanych. Pełny zestaw algebraicznych reguł transformacji drzewa pytania dla relacyjnych systemów rozproszonych można znaleźć w [11], a użycie wspólnych podwyrażeń pytań w celu uproszczenia drzewa pytania przedstawiono m.in. w [8]. Prezentacja wyników pytań nie stanowi większego problemu, gdyż można stosować dobre techniki implementacyjne opracowane dla scentralizowanych systemów ZBD.

Dekompozycja pytań globalnych na podpytania może się odbywać z użyciem transformacji równoważnościowych. Tego rodzaju transformacje w postaci algebry relacji kwalifikowanych (ang. qualified relations), zaproponowano w [7]. Relacja kwalifikowana jest uporządkowaną parą (R,qR), gdzie R jest relacją, zwaną ciałem relacji kwalifikowanej, a qR jest predykatem, zwanym kwalifikacją relacji kwalifikowanej. Algebra relacji kwalifikowanych jest rozszerzeniem algebry relacyjnej, w którym argumentami, oprócz zwykłych relacji, mogą być relacje kwalifikowane. Dla algebry relacji kwalifikowanych opracowano reguły wykonywania wszystkich operacji algebry relacyjnej. Na przykład, zastosowanie operacji selekcji S (R,F), gdzie R jest relacją kwalifikowaną (R,qR), a F jest predykatem selekcji, powoduje powstanie relacji z kwalifikacją *F AND qR*. Oznacza to, że w nowej relacji zachodzi zarówno *F* jak i *qR*. Koniunkcja predykatów powstaje także w operacji połączenia.

Należy zwrócić uwagę, że zastosowanie operacji selekcji lub połączenia może spowodować uzyskanie kwalifikacji sprzecznej. Na przykład, dla relacji *PRACOWNIK* może wystąpić sprzeczna kwalifikacja *DEPTNUM=1 AND DEPTNUM=5*. Relacje kwalifikowane o sprzecznych kwalifikacjach są naturalnie puste. Fakt ten może być rozpoznany już na etapie dekompozycji pytania, a nie dopiero podczas jego wykonania. Taka technika optymalizacji jest bardzo skuteczna, pod warunkiem jednak, że system umożliwia odpowiednie kwalifikowanie fragmentów.

Algebraiczna transformacja drzewa pytania oraz optymalizacja oparta na algebrze relacji kwalifikowanych mogą być stosowane w logicznej bazie danych i nie wymagają żadnych założeń dotyczących środowiska przetwarzania. Produkują one uproszczone drzewa pytania, które w procesie planowania pytań muszą być zamienione na sekwencje operacji, często nazywane programami pytań. Strategia przetwarzania pytania, którą wybierze moduł planujący, jest zdeterminowana przez algorytmy optymalizacji pytań.

Selekcja strategii przetwarzania wiąże się z następującymi problemami:
— określenie fizycznych kopii fragmentów, na których pytanie ma być realizowane;
— wybór kolejności wykonywania operacji;
— wybór metody wykonywania poszczególnych operacji.

Większość algorytmów optymalizacyjnych koncentruje się na dwóch ostatnich problemach. Ponieważ nie ma przeszkód, aby różne podwyrażenia wykonywały się na różnych kopiach tego samego fragmentu, kryterium lokalności (punkt pierwszy) rozstrzyga się na etapie selekcji kopii fizycznej.

Wybór kolejności wykonywania polega przede wszystkim na określeniu „najlepszej" sekwencji operacji połączenia, półpołączenia i sumy, ponieważ określenie kolejności wykonywania pozostałych operacji nie jest trudne. Mimo, że przetransformowane drzewo pytania reprezentuje częściowe uporządkowanie relacji, określenie kolejności połączeń w ramach tego samego poziomu drzewa nie jest problemem banalnym. Problem wyboru metody wykonywania operacji obejmuje wybór ścieżki dostępu fizycznego, a także „najlepszych" metod wartościowania połączeń.

Alternatywne strategie realizacji pytania, zarówno w środowisku scentralizowanym jak i rozproszonym, są wybierane na podstawie ich przewidywanej sprawności. Sprawność jest oceniana pod względem czasu odpowiedzi na pytanie (np. [1]) albo pod względem całkowitego czasu jego wykonania (np. [16]).

Jako przykład metody optymalizacji pytań w systemie o współdzielonych rozproszonych danych, warto przedstawić algorytm stosowany przez moduł planujący systemu NOMS [22]. Dane wejściowe dla modułu planującego stanowi algebraicznie przetransformowane drzewo pytania. Na wyjściu otrzymuje się kolekcję subprogramów pytań do wykonania w odpowiednich węzłach. W procesie planowania pytania są wykonywane następujące kroki:

Krok 1. Probabilistyczne szacowanie współczynników redukcji. Stwierdzenie zdolności redukcyjnych operacji algebry relacji oraz czynników jawnie redukujących (np. operacji półpołączenia i filtrów) ma wpływ na określenie modelu kosztów, a w konsekwencji na rezultaty algorytmu optymalizacji. Zastosowane podejście jest oparte na statystycznym modelu relacji schematu globalnego, zawierającym zarówno informację o mocy relacji, jak i parametry funkcji gęstości rozkładu wartości atrybutów. Dla dziedzin numerycznych stosuje się zwykłe rozkłady Gaussa, rozkłady Gaussa o wielu zmiennych losowych oraz rozkłady Beta, a dla dziedzin znakowych — ustalone selektywności ciągów znakowych.

Krok 2. Ustalenie kolejności połączenia dla każdego bloku pytania zawartego w drzewie pytania.

Obliczane są koszty użycia alternatywnych strategii implementacji operacji połączenia (dla każdej klasy operacji połączenia w systemie jest przechowywana kolekcja odpowiednich algorytmów). Rozpatrywane są wszystkie kombinacje par relacji. Drzewo połączeń, określające kolejność tych operacji, jest konstruowane według heurystycznego algorytmu modyfikacji poszczególnych rozgałęzień oraz algorytmu ograniczeń.

Krok 3. Wybór strategii implementacyjnych dla operacji teoriomnogościowych.

Strategie te są wybierane przez przeglądanie wszystkich możliwych w danym wypadku algorytmów realizacji danej operacji algebry relacyjnej.

Ponieważ wszystkie operacje elementarne zawarte w strategiach implementacji zdefiniowanych pierwotnie operacji relacyjnych są wyrażone w pośrednim języku manipulacji danymi, więc do oceny każdego z alternatywnych algorytmów można zastosować model kosztów. Selekcja konkretnego algorytmu oznacza jednocześnie określenie węzła dla wyników pośrednich. Oczywiście, znane są też węzły dla każdej relacji bazowej, zatem dekompozycja programu pytania na kolekcję subprogramów pytania odpowiadających każdemu węzłowi może być dokonana zupełnie prostym algorytmem.

STEROWANIE WSPÓŁBIEŻNOŚCIĄ W SYSTEMACH MULTIBAZ DANYCH

Zarządzanie rozproszonymi transakcjami wymaga rozwiązania kilku ściśle ze sobą powiązanych problemów, takich jak: niezawodność, efektywne użytkowanie zasobów całego systemu oraz sterowanie współbieżnością. Ostatniemu z wymienionych problemów, zwłaszcza dla systemów ZRBD, jest poświęcona obszerna literatura.

Podstawową cechą systemów multibaz danych jest fakt, że sprzęgają one lokalne systemy ZBD na poziomie użytkownika, tj. na poziomie języka pytań lub manipulacji danymi. Trudno jest więc zastosować algorytmy sterowania współbieżnością opracowane dla systemów ZRBD.

Brak współpracy między mechanizmami blokowania lokalnego i globalnego może na przykład spowodować powstanie niewykrywalnych globalnych zakleszczeń [12, 13].

Jednym z możliwych podejść do rozwiązywania tych problemów jest umożliwienie lokalnemu mechanizmowi sterowania współbieżnością wykrycia lokalnych zakleszczeń oraz unikanie globalnych zakleszczeń przez aborcje (ang. aborting) globalnie zakleszczonych transakcji z użyciem mechanizmu przeterminowania (ang. timeout). To podejście połączone z mechanizmem opóźnionej aktualizacji i odpowiednim protokołem kończenia (ang. commit protocol) może być wystarczające, o ile pogorszenie szybkości działania, spowodowane niepotrzebnym usuwaniem i wznawianiem transakcji, jest akceptowalne. Przykład takiego rozwiązania zaprezentowano w [21].

Mechanizmy sterowania współbieżnością, stosowane w systemach o rozproszonych współdzielonych danych, stwarzają projektantowi rozproszonych transakcji wiele ograniczeń. Przykładowo, w systemie NOMS ograniczenia te można wyrazić za pomocą następujących reguł:

- Każdy podproces rozproszonej transakcji (w tym proces główny) może wykonywać operacje tylko na jednej bazie danych.
- Z chwilą zakończenia przetwarzania lokalnej bazy danych, każdy z procesów podrzędnych musi wysłać do procesu głównego komunikat gotowy-do-zakończenia (ang. ready-to-commit).
- Wszystkie operacje aktualizacji, odnoszące się do lokalnej bazy danych mogą być zakończone dopiero po otrzymaniu komunikatu zakończenia od procesu głównego. W przeciwnym razie operacje aktualizujące są usuwane (na podstawie jawnego komunikatu o aborcji nadesłanego przez proces główny albo na skutek przeterminowania).
- Jeżeli w którymś z procesów podporządkowanych zdarzy się błąd, to do procesu głównego musi być wysłany komunikat aborcji. Dotyczy to wszystkich nienormalnych zakończeń transakcji, zarówno sterowanych przez programy aplikacyjne, jak i przez te programy systemowe, które oddają sterowanie do programów aplikacyjnych. W wypadku innych błędów, odpowiedni komunikat aborcji będzie generowany przez procedury sterujące systemu NOMS.
- W razie przeterminowania albo nadejścia komunikatu aborcji od jednego spośród procesów podrzędnych, proces główny musi przesłać komunikaty aborcji do wszystkich procesów podrzędnych.
- Proces główny może zakończyć swoje operacje aktualizujące oraz zakończyć rozproszoną transakcję tylko wówczas, gdy wszystkie procesy podrzędne tego procesu pomyślnie zakończyły przetwarzanie.

PROJEKTOWANIE ROZPROSZONYCH BAZ DANYCH

Projektowanie systemów RBD i systemów multibaz danych musi uwzględniać odmienną specyfikację tych dwóch rodzajów systemów. W wypadku środowiska o wielu bazach danych, w chwili tworzenia nowego zintegrowanego systemu przetwarzania informacji odpowiednie lokalne bazy danych już istnieją. Zwykle poważniejsza ich modyfikacja nie jest już możliwa. Systemy ZRBD są natomiast tworzone od zera (podobnie jak ich scentralizowane odpowiedniki) i pozostawiają wiele swobody dla decyzji projektanta.

Projektowanie RBD obejmuje wszystkie klasyczne kroki projektowania scentralizowanych baz danych, w tym fazy projektowania konceptualnego, logicznego i fizycznego. Etap logicznego projektowania rozproszonej bazy danych jest — w stosunku do jego scentralizowanego równoważnika — uzupełniony o projektowanie fragmentacji. Projekt fragmentacji określa sposób podziału relacji globalnych na: poziome, pionowe lub mieszane fragmenty [4, 5, 6, 9, 18, 19].

W projektowaniu fizycznym RBD występuje również etap projektowania alokacji fragmentów. Projektowanie alokacji fragmentów określa sposób odwzorowania fragmentów na obrazy fizyczne, a w konsekwencji także replikację fragmentów. Matematyczny problem optymalnej dystrybucji danych w sieci komputerowej, analizowano głównie w kontekście systemów rozproszonych plików (np. [10]), rzadziej natomiast w kontekście rozproszonych baz danych (np. [3]). Wyniki powyższych prac pomogły ustanowić kryteria projektowania alokacji, a także matematyczne podstawy dla potencjalnych narzędzi programowych, wspomagających projektantów RBD.

Projekt schematu globalnego dla systemu multibaz danych wymaga integracji i translacji modeli danych reprezentujących lokalne bazy danych. Dlatego proces integracji i abstrakcji jest bardziej ograniczony semantyką danych dostępnych w lokalnych bazach danych niż wymogami przetwarzania. Wyniki badań, które mogą być zastosowane do projektowania schematu globalnego w systemach multibaz danych, prezentowano m.in. w [17, 19]. Odwzorowania w heterogenicznym modelu danych, w szczególności wzajemne odwzorowanie między modelem sieciowym a relacyjnym, przedstawiono w [24].

Metodologie i narzędzia do projektowania RBD nie są jeszcze doskonałe i muszą być nadal rozwijane. Badania prowadzone w tej dziedzinie dają dobrą prognozę na przyszłość.

Kierunki rozwoju systemów o rozproszonych dzielonych danych mogą być przewidywane na podstawie projektów badawczych, prowadzonych w wiodących uniwersytetach i przemysłowych laboratoriach badawczych.

Wiele wysiłków badawczych poświęca się systemom tolerującym błędy (ang. fault tolerant systems), efektywnemu zarządzaniu transakcjami i mechanizmom odnawiania transakcji. Innym ważnym obszarem badawczym są inteligentne, przyjazne (ang. user-friendly) sprzężenia do systemów rozproszonych, pozwalające na użycie systemów o rozproszonych współdzielonych danych w systemach z bazą wiedzy.

Wydaje się, że systemy multibaz danych cieszą się w środowisku związanym z przetwarzaniem danych większym zainteresowaniem niż SZRBD. Jest to wynikiem silnej potrzeby integracji danych pod wpływem osiągniętego stopnia dojrzałości technik i narzędzi przetwarzania danych oraz — uświadomienia wymagań. Ogólna dostępność niezawodnych środków przesyłania danych, dla których granice państw nie stanowią żadnej przeszkody, wymusza rozwój adekwatnych narzędzi i technik.

Problemy związane z projektowaniem i użytkowaniem RBD stanowią znacznie większe wyzwanie niż analogiczne problemy zarządzania scentralizowaną bazą danych. Konieczne jest opracowanie odpowiednich narzędzi programowych, wspomagających procesy projektowania i użytkowania RBD. Obecne wyniki badań w tej dziedzinie są dość obiecujące.

Dotychczasowe techniki zarządzania RBD są prawie wyłącznie ukierunkowane na geograficznie rozproszone sieci. Obecnie, w okresie wzrastającego znaczenia sieci lokalnych LAN, wiele rezultatów opracowanych z myślą o systemach geograficznie rozproszonych będzie z pewnością transplantowanych do tego środowiska. Ponadto w lokalnych sieciach typu gwiaździstego, dla niektórych problemów, jak na przykład dla problemu przetwarzania pytań, najprawdopodobniej uda się znaleźć bardziej efektywne rozwiązania.

Można przypuszczać, że wzrastający nacisk na funkcjonalną integrację logicznie i geograficznie rozproszonych sieci będzie silnym i długotrwałym bodźcem do intensywnego rozwoju całej problematyki systemów o rozproszonych współdzielonych danych.

Tłumaczył i opracował:
WIKTOR RZECZKOWSKI

LITERATURA

[1] Brill D. et al.: Distributed Query Optimization in MERMAID — A Frontend to Data Management Systems. Proc. Int. Conf. on Data Engineering, Los Angeles, April 1984

[2] Breitbart Y. J., Tieman L. R.: ADDS — Heterogeneous Distributed Database System. Proc. 3rd Int. Seminar on Distributed Data Sharing Systems, Parma, Italy, March 1984; F. A. Schreiber and W. Litwin (eds.), North Holland, Amsterdam, 1985

[3] Ceri S. et al.: Optimal File Allocation in a Computer Network — A Solution Method on the Knapsack Problem. Computer Networks, Vol. 6, No. 5, 1982

[4] Ceri S. et al.: Horizontal Partitioning in Database Design. Proc. ACM SIGMOD'82 Int. Conf. on Management of Data, 1982

[5] Ceri S., Navathe S. B.: A Methodology for the Distribution Design of Databases. Proc. COMPCON'83 Conf., San Francisco, March 1983

[6] Ceri S. et al.: Distribution Design of Logical Database Schemes. IEEE Trans. on Software Engineering, Vol. SE-9, No. 4, 1983

[7] Ceri S., Pelagatti G.: Corectness of Query Execution Strategies in Distributed Databases. ACM Trans. on Database Systems, Vol. 8, No. 4, 1983

[8] Ceri S., Pelagatti G.: Distributed Databases — Principles and Systems. McGraw-Hill, New York, 1984

[9] Davenport R. A.: Design of Distributed Data Base Systems. Computer Journal, Vol. 24, No. 1, 1981

[10] Dowdy L. W., Foster D. V.: Comparative Models of the File Assignment Problem. ACM Computing Surveys, Vol. 14, No. 2, 1982

[11] Forker H. J.: Algebraical and Operational Methods for the Optimization of Query Processing in Distributed Relational Database Systems. Proc. 2nd Symposium on Distributed Data Bases, Berlin, 1982. H.-J. Schneider (ed.), North-Holland, Amsterdam, 1982

[12] Gligor V. D., Luckenbaugh G. L.: Interconnecting Heterogeneous Database Management Systems. Computer, Vol. 17, No. 1, 1984

[13] Gligor V. D., Popescu-Zeletin R.: Concurrency Control Issues in Distributed Heterogeneous Database Management Systems. Proc. 3rd Int. Seminar on Distributed Data Sharing Systems, Parma, Italy, March 1984; F. A. Schreiber, W. Litwin (eds), North Holland. Amsterdam, 1985

[14] Landers T., Rosenberg R. L.: An Overview of MULTIBASE Proc. 2nd Int. Symposium on Distributed Data Bases. H.-J. Schneider (ed.), North-Holland, Amsterdam, 1982

[15] Litwin W. et al.: SIRIUS Systems for Distributed Data Management. Proc. 2nd Int. Symposium on Distributed Data Bases. H.-J. Schneider (ed.), North Holland, Amsterdam, 1982

[16] Lohman G. M. et al.: Query Processing in R*. W. Kim et al. (eds.), Query Processing in Database Systems. Springer-Verlag, Berlin, 1984

[17] Navathe S. B., Gadgil S.: A Methodology for View Integration. Proc. 8th Int. Conf. Very Large Databases, Mexico City, September 1982

[18] Navathe S. B. et al.: Vertical Partitioning Algorithms for Database Design. ACM Trans. on Database Systems. Vol. 9, No. 4, 1984

[19] Navathe S. B. et al.: Relationship Merging in Schema Integration. Proc. 10th Int. Conf. Very Large Databases, Singapore, 1984

[20] Smith J. M. et al.: MULTIBASE — Integrating Heterogeneous Distributed Database Systems. Proc. AFIPS National Computer Conference, Vol. 50, 1981

[21] Staniszkis W. et al.: Architecture of the Network Data Management System. Proc. 3rd Int. Seminar on Distributed Data Sharing Systems. Parma, Italy, March 1984; F. A. Schreiber, W. Litwin (eds.), North Holland, Amsterdam, 1985

[22] Staniszkis W. et al.: NOMS Query Planer; W. Staniszkis (ed.), NOMS Working Papers. CRAI Research Report 84-24, Rende (Italy), October 1984

[23] Tempelton M. et al.: An Overview of the MERMAID System — A Frontend to Heterogeneous Databases. Proc. 18th Annual Electronics and Aerospace Conf., Washington (D.C.), 1984

[24] Zaniolo C.: Design of Relational Views over Network Schemes. Proc. ACM SIGMOD'79 Int. Conf. on Management of Data, 1979.

Konferencje

INFOGRYF '88

W dniach 18—21 października 1988 r. odbędą się kolejne **IX Kołobrzeskie Dni Informatyki INFOGRYF '88** pod tradycyjnym już hasłem „Informatyka w zarządzaniu przedsiębiorstwem". Hasło jest tradycyjne, lecz ta konferencja będzie poświęcona systemom informatycznym zarządzania nowej generacji. Jej organizatorami są TNOiK (oddział w Szczecinie), Instytut Cybernetyki Ekonomicznej i Informatyki Uniwersytetu Szczecińskiego oraz ZETO Szczecin i Koszalin.

INFOGRYF będzie organizowany w pięciu sesjach:

- Systemy informatyczne zarządzania (struktury — funkcjonowania — rozwój) — sekretarz dr E. Kolbusz i dr E. Kram;
- Systemy wspomagania decyzji (istota — zasady — możliwości zastosowań) — sekretarz doc. dr hab. A. Nowakowski i dr Z. Drążek (sesja symulacji);
- Metoda tworzenia i rozwoju systemów (stan i perspektywy rozwoju — zastosowania praktyczne) — sekretarz dr J. Marcinkiewicz;
- Mikroskala'88 (wersje i kontrowersje zastosowań mikrokomputerów) — sekretarz dr Z. Szyjewski;
- Giełda systemów i programów — sekretarz dr J. Kowalska.

Celem wskazania autorom tematyki preferowanej przez organizatorów podajemy dla ukierunkowania referatów słowa kluczowe dziedzin: nowe generacje systemów informatycznych, systemy informowania, bazy danych, systemy wspomagania decyzji, bazy wiedzy, systemy wiedzy, systemy ekspertowe, logistyka, sztuczna inteligencja, bazy metod i modeli, symulacja, systemy „Just-in-time", projektowanie, dokumentowanie systemów i programów, inżynieria programowania, inżynieria systemów, integracja rozwiązań systemowych.

Nie jest to zbiór pełny i organizatorzy obiecują rozpatrzyć każde zgłoszenie referatu.

Adres organizatorów:
Towarzystwo Naukowe Organizacji i Kierownictwa Oddział w Szczecinie
ul. Pocztowa 30/12, 70-360 Szczecin

KAZIMIERZ BIEŃKOWSKI
Instytut Informatyki
Politechnika Warszawska

Systemy przetwarzania tekstów

Systemy przetwarzania tekstów (SPT) są informatycznymi systemami cyfrowymi typu dialogowego, w których udział człowieka-operatora decyduje o powodzeniu przetwarzania. Podobnie jak przetwarzanie danych, rozpoczyna się od wprowadzenia ich do komputera, przetwarzanie tekstów rozpoczyna się od wprowadzania słów.

Maszynistka pisze ze średnią szybkością 250 znaków/minutę i robi jeden błąd średnio co 600 znaków. Na poprawienie tego błędu, na oryginale i dwu kopiach pisanych na zwykłej maszynie do pisania, traci ona średnio 90 sekund. Takie samo poprawienie z wykorzystaniem SPT zajmuje tylko 5 sekund.

Średnia szybkość zapisu odręcznego, zwykłego wynosi 10 słów na minutę. Skuteczna szybkość dyktowania stenotypistce wynosi 30 słów na minutę. Może ona w zasadzie stenografować jeszcze szybciej, jest jednak uzależniona od przerw, które robi autor dyktujący tekst oraz ona sama. Największą szybkość notowania wypowiedzi rzędu 50 słów na minutę, osiąga się przy notowaniu na dyktafon.

Wydajność maszynopisania z tych trzech źródeł informacji zapisanej na nośnik wynosi odpowiednio:

- dla rękopisu zwykłego — 10 słów na minutę,
- dla stenogramu — 15 słów na minutę,
- dla dyktafonu — 20 słów na minutę.

Kombinując odpowiednie czasy zapisu tekstu mówionego z czasami jego odczytu i maszynopisania, otrzymuje się odpowiednio wydajności pracy:

- dla rękopisu — 60 wierszy na godzinę,
- dla stenogramu — 60 wierszy na godzinę,
- dla dyktafonu — 120 wierszy na godzinę.

Wykorzystując SPT można tę wydajność znacznie polepszyć.

STRUKTURA SYSTEMÓW PRZETWARZANIA TEKSTÓW

Z pewnym uproszczeniem można wprowadzić sześć klas zadań dla SPT:

1. Pojedynczy krótki list;
2. Krótki list standardowy, powielany w wielu kopiach;
3. Pojedynczy długi list, mały raport (z niewielką korektą);
4. Pojedynczy długi list, mały raport (z intensywną korektą);
5. Długi raport, złożony harmonogram (z korektą);
6. Długi dokument złożony ze standardowych paragrafów, złożone sortowanie, listy itp.

Doc. dr hab. inż. KAZIMIERZ BIEŃKOWSKI jest docentem etatowym w Instytucie Informatyki Wydziału Elektroniki Politechniki Warszawskiej. Doktoryzował się w zakresie badań niezawodnościowych ferrytowych pamięci operacyjnych komputerów. Jego rozprawa habilitacyjna była poświęcona metodom badań niezawodnościowych cyfrowych układów scalonych, w tym konstrukcji specjalizowanych systemów komputerowych do takich badań. Obecnie specjalizuje się w zakresie technik mikrokomputerowych oraz w zagadnieniach ergonomiki informatycznej.

Ze względu na strukturę logiczną i budowę części elektronicznej i elektromechanicznej, można wyróżnić cztery klasy elektronicznych SPT:

klasa A — zwykła maszyna pisząca (mechaniczna, elektromechaniczna, elektroniczno-mechaniczna),
klasa B — autonomiczny (bez telekomunikacji) SPT, bez monitora ekranowego,
klasa C — autonomiczny (bez telekomunikacji) SPT z monitorem ekranowym,
klasa D — wielodostępny (ang. shared logic) SPT z łączami telekomunikacyjnymi.

Uwzględniając klasyfikację zadań przetwarzania tekstów oraz klasy A, B, C, D (SPT), zwykle proponuje się użytkownikom następujący dobór sprzętu do zadań:

Klasa zadania	Typ SPT
1	A
2	B
3	B,C
4	C,D
5	C,D
6	D

Zwykle zachodzi jednak potrzeba dokładniejszego określenia, zarówno cech zastosowań (zadań), jak i optymalnych cech sprzętu SPT.

CECHY JEDNOSTEK PISZĄCYCH

Wybór rodzaju drukarki, ze względu na jej cechy fizyczne i funkcjonalne, jest dość krytyczny i ściśle uzależniony od przewidywanych zastosowań. Pod tym względem współczesne SPT znacznie różnią się od komputerów klasycznych, w których typ drukarki jest zwykle mało krytyczny, a w szczególnych wypadkach stosowanie drukarki jest zbędne.

Wyróżnia się trzy klasy jakości tekstu drukowanego. Inną cechą doskonałości wydruków jest ich wyrazistość (inaczej „ostrość").

Jakość poligraficzna

Tekst drukowany o jakości poligraficznej (ang. graphic art quality) jest w zasadzie przeznaczony dla maszyn poligraficznych do fotoskładu i wymaga złożonego formatowania oraz oprogramowania do cytowania (ang. hyphenation) i korygowania. Zmienne są zarówno kroje czcionek jak i ich rozmiary. Wydruki tego rodzaju są potrzebne wyłącznie w biurach przygotowujących teksty w celu ich rozpowszechniania poligraficznego. Wydruk o jakości poligraficznej jest często wydrukiem barwnym. Wtedy cechą jego każdego punktu (ang. pixel) jest liczba atrybutów (m.in. kolorów) przypadających na ten punkt (patrz [1]).

Jakość korespondencyjna

Teksty o jakości korespondencyjnej (ang. correspondence quality) są przeznaczone do korespondencji zewnętrznej firmy i krótkich raportów, w celu dystrybucji poza obrębem danego biura. W zakresie tym obserwuje się tendencje do formatowania, stosowania czcionki proporcjonalnej i niekiedy również oprogramowania korekcyjnego. Zwykle dysponuje się wieloma krojami czcionek, jednak przy zachowaniu jednego ich wymiaru. Tej klasy jakości zwykle oczekuje się w większości biur administracyjnych.

Jakość szkicowa

Jakość szkicowa (ang. draft quality) jest równoważna wydrukowi z dalekopisu lub z komputera i zwykle uznaje się ją za wystarczającą dla dokumentów wewnętrznych. Rzadko stosowane jest niewielkie formatowanie. Zwykle stosuje się również tylko jeden krój czcionek, ze stałą po-

działką 10 do 12 znaków na cal. Przy udoskonaleniach merytoryczności i czytelności dopuszcza się tę jakość dla nieprestiżowego i ekonomicznego obiegu korespondencji zewnętrznej.

Wyrazistość wydruku

Wyrazistość wydruku, zwaną inaczej zdolnością rozdzielczą, określa się zwykle podając liczbę drukowanych linii na milimetr, bądź też liczbę wydrukowanych punktów na cal (ang. points per inch, pts/in, lub pixels per in, pxs/in). Jest to parametr niezależny od liczby krojów czcionek i może być określany nawet dla drukarek numerycznych, dysponujących tylko 10 czcionkami cyfr o jednym kroju.

RODZAJE DRUKAREK

We współczesnych SPT stosuje się następujące rodzaje drukarek:

- czcionkowe (ang. typebar),
- z głowicą czcionkową (ang. golfball),
- płatkowe (ang. petal printer, daisywheel),
- mozaikowe (ang. dot matrix),
- bezuderzeniowe (ang. non-impact printer),
- z natryskiem ciągłym kropel atramentu (ang. continuous jet, ink jet),
- z pulsowym natryskiem kropel atramentu (ang. impulse jet),
- kserograficzne (ang. electrographic printers),
- laserowe kserograficzne (ang. laser electrographic printers),
- elektrostatyczne kserograficzne (ang. electrostatic printers).

Drukarki czcionkowe

Drukarki czcionkowe należą do klasy drukarek uderzeniowych (ang. impact printers), przeznaczonych do pisania na zwykłym papierze (ang. plain paper printers).

Zasada pisania polega na tym, że ruchoma dźwignia, o względnie dużej długości ramienia, z umieszczoną na jej końcu czcionką drukarską, uderza poprzez taśmę barwiącą (kopiującą) w zwykły papier nawijany na wałek gumowy.

Ze względu na znaczną energię uderzenia możliwe jest jednoczesne nawinięcie na wałek kilku warstw papieru piszącego i kalki kopiującej. Zezwala to na jednoczesne drukowanie wielu kopii węglowych (zwykle 4—5), co prawda o mniejszej wyrazistości. Przy zastosowaniu specjalnych papierów, na przykład bardzo cienkich (bibułowych) lub powlekanych warstwą kalkującą (kopiującą) na odwrocie, liczba kopii może być zwiększona nawet do 10—12. Ponadto, ze względu na dużą siłę uderzenia, na drukarki tego typu można zakładać matryce metalowe lub woskowe, przeznaczone do powielania — zwykle pozwalające na druk kilkuset kopii oryginału, aż do zniszczenia (zamazania) matrycy.

Czcionka drukarki tego typu jest zwykle dwupolowa górna (ang. upper case) i dolna (ang. lower case). Drukuje ona znak bądź z pola górnego (np. małe litery), bądź z pola dolnego (duże litery), dzięki dodatkowemu ruchowi pionowemu, bądź wałka gumowego, bądź zespołu dźwigni czcionkowych.

Ze względu na konstrukcję mechanizmu przesuwu wałka gumowego względem zespołu czcionek, wyróżnia się:

- drukarki czcionkowe z ruchomym wałkiem (do klasy tej należą wszystkie mechaniczne, niezelektryfikowane maszyny piszące),
- drukarki czcionkowe z ruchomym wałkiem czcionek (do klasy tej należą niektóre typy dalekopisów i elektromechanicznych drukarek czcionkowych); ten typ stosowany jest przy bardzo szerokich (długich) wałkach, z możliwością wymiany wałka.

Skuteczna szybkość pisania dla drukarek czcionkowych wynosi 10—15 znaków na sekundę, przy standardowej szybkości 6,8 znaku na sekundę dla dalekopisu.

Elektromechaniczne drukarki czcionkowe piszące w trybie automatycznym są szczególnie dokuczliwe dla otoczenia, głównie ze względu na znaczny poziom hałasu. Ponadto maszyny automatyczne z ruchomym wałkiem mogą być przyczyną obrażeń fizycznych (zranień) personelu, przy nieoczekiwanym gwałtownym ruchu wałka.

Dodatkową wadą maszyn tej klasy jest ich niski poziom niezawodności i mała trwałość, zwykle zadowalające tylko wtedy, gdy z maszyny korzysta się sporadycznie lub rzadko.

Drukarki głowicowe

W drukarkach tego typu zespół czcionek jest połączony trwale w jeden element mechaniczny. Wyróżnia się przy tym następujące rodzaje zespołów czcionkowych:

— zbiór listew drukujących (ang. line printer),

— łańcuch drukujący (ang. chain printer),

— wirujący bęben drukujący (ang. drum printer, on-the-fly printer),

— obracający się walec drukujący, obracająca się kula drukująca (ang. golfball printer).

Trzy pierwsze typy drukarek głowicowych są zwykle nazywane żargonowo drukarkami wierszowymi, gdyż zwykle drukują one jednocześnie cały wiersz strony. Wydajność drukowania dla drukarek tego typu jest znaczna i wynosi zwykle od 5 do 30 wierszy na sekundę, przy liczbie znaków w wierszu dochodzącej do 256 i szybkości przesuwu papieru do 1,5 metra na sekundę. Ze względu na znaczną szybkość drukowania zachodzi tu potrzeba stosowania specjalnego papieru, obrzeżnie perforowanego (tak zwany napęd Leporello papieru), z wykluczeniem napędu papieru wałkiem gumowym. Ponadto zachodzi potrzeba automatycznego podawania i składania papieru. Drukarki tego typu są szeroko stosowane w technice komputerowej, począwszy od 1960 roku. W SPT są one rzadko stosowane, zwykle przy masowym generowaniu powtarzalnych wydruków o szkicowej jakości drukowania lub w wielodostępnych SPT, jako monitorowe drukarki systemowe. W drukarkach tego typu jest możliwe generowanie niewielkiej liczby kopii (2—3), przy zastosowaniu papieru specjalnego z warstwą kopiującą na odwrocie. Przenoszenie wydruku na oryginał odbywa się z taśmy kopiującej, jednak o jakości wyższej niż wymagana dla maszyn czcionkowych, stylonowej, ewentualnie z odświeżaniem czynnika kopiującego (atramentu). Efektywna szybkość drukowania dla drukarek wierszowych może dochodzić do 5000 znaków na sekundę, a zwykle wynosi od 600 do 1800 znaków na sekundę i może być większa tylko dla drukarek kserograficznych lub piszących na nośnikach specjalnych (nie na papierze).

W stosunku do innych typów drukarek, liczba krojów (typów) czcionek jest ograniczona: od 10 (w drukarkach numerycznych z listwami drukującymi) do 64 (w drukarkach bębnowych), rzadko do 128 (w drukarkach łańcuchowych). Nie zachodzi tu również praktycznie możliwość zmiany krojów czcionek, podobnie jak w maszynach czcionkowych, przy względnie niskiej jakości wydruku (niższej ze względu na wzrost szybkości w stosunku do zwykłych maszyn czcionkowych).

Drukarki z obracalną głowicą drukującą

W stosunku do opisanych wcześniej drukarek liniowych, drukarki z obracalną głowicą drukującą mają następujące cechy:

— głowica drukująca sztywna, zwykle o kształcie wycinka kuli lub walca obraca się tylko podczas zmiany drukowanego znaku, a nie wiruje w sposób ciągły,

— na powierzchni kuli lub walca drukującego mieści się zwykle 96 znaków w czterech wierszach,

— efektywna szybkość drukowania dla drukarek tego typu wynosi 10—20 znaków na sekundę, co jest porównywalne z szybkością mechanicznej maszyny piszącej,

— trwałość obracalnej sztywnej głowicy piszącej jest porównywalna z trwałością zespołu czcionkowego mechanicznej maszyny piszącej,

— obracalna głowica pisząca jest zwykle kształtką plastykową z metalizowaną powierzchnią piszącą, co warunkuje jej niski koszt (rzędu 25 dolarów) i możliwość dysponowania wymiennym zestawem krojów czcionek (głowic), co jest niemożliwe do osiągnięcia w mechanicznych maszynach piszących.

Typowa elektryczna maszyna pisząca z głowicą obracalną IBM Selectric dysponuje zestawem 30 wymienialnych głowic, w tym głowicą o kroju czcionek odczytywalnym przez optyczne czytniki pisma (ang. optical character recognition, OCR).

Zwykle dodatkowym wyposażeniem drukarek tego typu jest dwubarwna taśma pisząca oraz taśma kasująca zapis przy cofaniu podczas korekty, co jest możliwe ze względu na mały ciężar i gabaryty zespołu drukującego.

Drukarki płatkowe

Drukarki płatkowe, zwane też rozetkowymi, mają konstrukcję podobną do drukarek z obracalną głowicą drukującą, w tym sensie że:
— wózek drukarki jest nieruchomy w poziomie, zaś ruch poziomy wykonuje tylko głowica płatkowa (ang. daisy-wheel),
— oprócz ruchu poziomego, odpowiadającego pozycji w wierszu drukowanego znaku, głowica płatkowa wykonuje ruch obrotowy w celu selekcji odpowiedniej pary znaków umieszczonej na płatku, podobnie jak na dźwigni maszyny czcionkowej.

Głowica płatkowa jest jednak tylko częściowo sztywna; czcionki drukujące są umieszczone na jej rozciętym obwodzie i mogą się uginać pod uderzeniami młotka drukującego, odbijając za pośrednictwem taśmy barwiącej odpowiedni znak (górny lub dolny) na papierze.

Ze względu na małą wagę jednostki piszącej stosuje się tu również wielobarwne taśmy barwiące oraz wycierające taśmy korekcyjne. Głowice płatkowe mogą być również wymienialne (dostępne są zestawy wymienne krojów czcionek). Są one tańsze od sztywnych głowic obracalnych (koszt rzędu 15 dolarów), jednak ich trwałość jest dość niska (od 500 do 1000 godzin pracy ciągłej).

Ze względu na małą wagę głowicy piszącej, drukarki płatkowe piszą ze standardową szybkością 25 znaków na sekundę do 55 znaków na sekundę w wykonaniu szybkim. Charakteryzują się one małą hałaśliwością pracy.

Drukarki mozaikowe

Nazwą drukarek mozaikowych określane są mechaniczne drukarki uderzeniowe, piszące na zwykłym papierze przez taśmę barwiącą, za pośrednictwem zestawu igieł drukujących. Liczba igieł waha się od 7 do 24, przy rozdzielczości dochodzącej do 150 punktów na cal. Drukarki te piszą z szybkościami od 50 do 360 znaków na sekundę.

W stosunku do poprzednio opisanych, drukarki mozaikowe wyróżniają się większą szybkością drukowania, małą hałaśliwością oraz możliwością dynamicznej (elektronicznej) zmiany kroju drukowanych znaków.

Tendencje unowocześniania drukarek mozaikowych dotyczą zastosowania dodatkowego mikroprocesorowego sterowania ruchem wałka niosącego papier, w celu podwyższenia rozdzielczości, przy liczbie igieł ograniczonej do 7 (drukarki amerykańskie firmy Sanders).

Drukarki bezuderzeniowe

Drukarkami bezuderzeniowymi określa się wszystkie jednostki piszące na papierach specjalnych. Rozróżnia się trzy rodzaje papierów specjalnych:
— do zapisu termicznego (ang. thermal printer),
— do zapisu magnetycznego (ang. magnetic printer),
— do zapisu przy użyciu pola elektrycznego (ang. electrosensitive printer).

Ze względu na prostotę części mechanicznej oraz niski pobór energii, drukarki bezuderzeniowe są stosowane chętnie w przenośnych urządzeniach drukujących; na przykład w kalkulatorach z zapisem lub w przenośnych terminalach sieci komputerowych.

Dwiema poważnymi wadami tych drukarek są:
— konieczność posiadania odpowiednich zapasów papieru specjalnego,
— możliwość przypadkowego zniszczenia zapisanego dokumentu.
Eliminują one praktycznie ten typu drukarek z zastosowań w systemach przetwarzania tekstów.

Szybkość zapisu i zdolność rozdzielcza tych drukarek są takie jak dla drukarek mozaikowych. Zwykle również kształt znaku jest określony przez mozaikę punktów, z możliwością dynamicznej zmiany zasobu znaków, oraz z możliwością stosowania wyjść graficznych.

Drukarki z ciągłym natryskiem kropel atramentu

Drukarka ta pracuje na zasadzie ciągłego natrysku atramentu, który dzięki wibracjom ultradźwiękowym jest rozrywany na krople o stałej objętości. Krople te uzyskują następnie pewien ładunek elektryczny, dzięki któremu mogą być odchylane elektrostatycznie (tak jak elektrony w promieniowej lampie elektronowej) w kierunku papieru, na który nanoszą krótkie szeregi kropek. W trakcie przemieszczania wózka drukującego kropki są rozmieszczane sekwencyjnie obok siebie, formując elementy kolejnych drukowanych znaków.

W technice tej osiąga się rozdzielczość do 240 punktów na cal i — mimo mozaikowej zasady wydruku — czytający nie może w pierwszej chwili wyróżnić elementów dyskretnych mozaiki.

Dla założonej korespondencyjnej jakości wydruku osiąga się szybkości do 100 znaków na sekundę. Przy użyciu matrycy 5×7 dysz można zwiększyć tę szybkość do 1500 znaków na sekundę.

Przykładową maszyną w tej klasie jest drukarka firmy IBM typu 6640. Ma ona szybkość podstawową 92 znaki na sekundę, pracuje cicho, dysponując zestawem pięciu elektronicznie zmienialnych krojów czcionek. Ostatnio szybkość jej wydruku zwiększono do 200 znaków na sekundę.

Przy jakości szkicowej osiąga się szybkości do 300 znaków na sekundę. Przy zastosowaniu tablicy dysz na cały wiersz szybkość wydruku można zwiększyć do 14 stron na sekundę, z rozdzielczością 120 punktów na cal.

Drukarki z impulsowym natryskiem atramentu

Drukarki z impulsowym natryskiem atramentu mają zawory natryskujące atrament w sposób dyskretnie sterowalny. Pierwszym urządzeniem handlowym tego typu są drukarki mozaikowe o jakości szkicowej. Obecne szybkości wydruku dochodzą do 300 znaków na sekundę, jednak notuje się tu szybki postęp zarówno w szybkości, jak i jakości wydruku.

Pierwszymi drukarkami rynkowymi tego typu są modele Siemens PT80 oraz Siliconics Quietype. Firma Olympia zademonstrowała również prototyp o jakości korespondencyjnej.

Firma Siliconics twierdzi, że po udoskonaleniach technologicznych uda się, przy umiarkowanych szybkościach wydruku i cenie, osiągnąć zdolność rozdzielczą 200—500 punktów na cal [1]).

Drukarki kserograficzne

Drukarki kserograficzne pracują na zasadzie podobnej do pracy kopiarek kserograficznych z selenowymi płytami do przenoszenia obrazu optycznego, to jest na zasadzie tak zwanej suchej fotografii.

W metodzie suchej fotografii (kserokopii) konieczne jest wytworzenie optyczne drukowanego tekstu, który następnie jest projektowany (rzutowany) na powierzchnię wirującego ze stałą szybkością bębna selenowego. Rzutowany obraz optyczny powoduje rozładowanie powierzchni bębna uprzednio naładowanej stałym ładunkiem, za pomocą źródła wysokiego napięcia. Obraz ładunków z bębna selenowego jest następnie „wywoływany", przez posypanie wirującej powierzchni proszkiem barwiącym. Z wirującego bębna proszek barwiący jest przenoszony, dzięki przyciąganiu elektrostatycznemu, na papier (zwykły, o kontrolowanych niektórych cechach). Papier z przyciągniętym obrazem proszkowym przechodzi przez strefę utrwalania, chemicznego lub termicznego, gdzie proszek jest trwale spajany z powierzchnią papieru. Zapis tego rodzaju jest trwały, charakteryzuje się ostrością konturów lepszą niż przy zapisie uderzeniowym i niezbyt dobrą czernią pól o dużej powierzchni (ze względu na silne odpychanie się jednoimiennych ładunków). Resztki proszku z bębna, po przeniesieniu na papier, są zgarniane specjalną nieruchomą szczotką, tak że bęben jest gotów do następnej operacji przed zakończeniem każdego kolejnego obrotu.

[1]) Zwykła mechaniczna maszyna pisząca dobrej jakości, z taśmą karbonizowaną, daje rozdzielczość do 300 punktów na cal.

Ze względu na przyjętą zasadę pracy, drukarki kserograficzne charakteryzują się bardzo dużą szybkością drukowania, ograniczoną tylko dopuszczalną szybkością liniową taśmy papierowej i „męczeniem się" powierzchni selenu. Można osiągnąć szybkość 1—2 metrów na sekundę, przy zdolności rozdzielczej rzędu 300 punktów na cal (w stosunku do konturów znaków). Tak ogromna szybkość drukowania jest często niemożliwa do wykorzystania w praktyce. Praktycznie osiąga się szybkość 0,2—0,5 strony na sekundę.

W drukarkach kserograficznych często komponuje się tekst z monitora komputerowego z obrazem rzutowanym na bęben selenowy z mikrofilmu, którym mogą być:
— teksty stałe,
— rysunki i ilustracje,
— formularze,
co daje znaczny wzrost efektywności zapisu w stosunku do drukarek innych typów.

Jako źródło tekstu zmiennego, generowanego przez komputer, są wykorzystywane:
— lampa błyskowa,
— lampy elektronopromieniowe (monitory ekranowe),
— diody LED,
— bramki optyczne.

Drukarki kserograficzne są znane[2]) i produkowane od 1960 roku, z pionierskimi pracami angielskiej firmy Rank Xerox.

Laserowe drukarki kserograficzne

Laserowe drukarki kserograficzne pracują na zasadzie podobnej do opisanych poprzednio, przy czym źródłem obrazu generowanego przez komputer jest odchylana wiązka laserowa. Stosuje się odchylanie tej wiązki za pomocą wirującego systemu zwierciadeł (wirujący wielobok płaskich zwierciadeł) lub za pomocą modulatorów elektrooptycznych.

Drukarki laserowe drukują z rozdzielczością do 10 linii na milimetr, przy wydajności 15—20 000 wierszy na minutę lub maksymalnie 2 stron formatu A4 na sekundę. Typowymi drukarkami tej klasy są urządzenia:
— Hewlett Packard 2680,
— IBM 3800,
— Xerox 9700,
— Siemens 3352.
Sądzi się, że po zastąpieniu bębna selenowego przez bęben z fotoczułego plastyku, trwałość bębna aktywnego wzrośnie do 20 tysięcy kopii.

Elektrostatyczne drukarki kserograficzne

Elektrostatyczne drukarki kserograficzne stanowią odmianę drukarek kserograficznych, bez wirującego bębna selenowego.

Ładunek zapisywany jest przenoszony na zwykły papier pokrywany plastykiem[3]) bezpośrednio z grzebienia elektrod ostrzowych, wywoływany metodą proszkową i utrwalany, podobnie jak w drukarkach kserograficznych, dając zapis typu mozaikowego.

Brak bębna selenowego, eliminujący „męczenie się" klisz selenowych, daje znaczne uproszczenie konstrukcji. Inne cechy są podobne jak dla drukarek mozaikowych, bez możliwości mechanicznego generowania kopii. Rozdzielczość wynosi 160—300 punktów na cal. W drukarkach tego typu osiąga się szybkość drukowania od 0,1 do 0,5 strony A4 na sekundę. Elektrody piszące są wybierane selektywnie impulsami rzędu 500 V, z lekkim wyładowaniem koronowym.

* * *

Można wyróżnić osiem klas zastosowań systemów przetwarzania tekstów:
1) korespondencja zwykła (ang. general correspondence),
2) korespondencja intensywna (ang. repetitive correspondence),
3) redagowanie,
4) produkcja dokumentów,
5) księgowość,
6) dokumentacja techniczna,
7) wypełnianie formularzy,
8) obsługa list i wykazów.

W tabeli podano zestaw pytań decyzyjnych, ułatwiających sformułowanie wykazu cech funkcjonalnych SPT przewidzianych do danych zastosowań. Wykaz ten może być podstawą w wyborze jednego z wielu istniejących SPT lub w opracowaniu własnego. Określenia zestawu cech funkcjonalnych SPT, spełniających wymagania wynikające z zastosowań, można dokonać na podstawie podanej listy pytań.

Lista pytań decyzyjnych ułatwiających wybór SPT do określonych zastosowań

Cechy ogólne SPT:
- pojemność pamięci zewnętrznej,
- struktura i pojemność pamięci operacyjnej mikrokomputera,
- liczba drukarek terminalowych (operatorskich),
- typ i pojemność systemu pamięci buforowej,
- wyrównanie prawych marginesów przez zmianę odstępów między słowami i znakami,
- czy dokument ma być tworzony przez scalanie dwu lub więcej dokumentów zapamiętywanych na nośniku,
- czy przy scalaniu dokumenty pierwotne są zachowywane,
- maksymalna liczba stacji drukujących (w wersji wielodostępnej),
- możliwości sumowania i innych operacji matematycznych,
- cena,
- koszt obsługi,
- koszt dostawy,
- okres gwarancyjny,
- koszt szkolenia operatorów,
- jakość instrukcji (podręcznika obsługi) dla operatorów,
- możliwość samouczenia się,
- rodzaj nośnika pamięci zewnętrznej (karty magnetyczne, dyskietki, pamięci dyskowe, kasety taśmowe),
- liczba dodatkowych kluczy sterujących (przycisków),
- czy występuje niezależna klawiatura numeryczna (ang. numerical pad),
- możliwości wielodostępu,
- zabezpieczenia systemowe, hasła blokujące,
- pamięci formatów, formaty wielokrotne,
- wprowadzanie formularzy,
- ergonomiczność systemu,
- wbudowywalność do mebli.

Cechy podsystemu monitorowego:
- parametry wymiarowe monitora (liczba znaków w wierszu, liczba wierszy),
- geometryczna wielkość znaków,
- czy występuje dynamicznie sterowalne zwiększanie lub zmniejszanie znaków,
- przeglądanie (ang. scrolling) poziome i pionowe,
- ograniczenia dla przeglądania,
- komunikaty oraz polecenia systemowe,
- pokazywanie podkreśleń,
- pokazywanie zmian wysokości czcionek (indeksów i stopek),
- kolor ekranu,
- pokazywanie wytłuszczeń i zmian stylu (kroju) czcionek,
- typ kursora,
- ograniczenia ruchu kursora,
- czy jest opcja proporcjonalności,
- czy tekst jest wprowadzany na ekran od dołu czy od góry,

[2]) W latach sześćdziesiątych prace w dziedzinie kopiarek kserograficznych i kserograficznych wyjść komputerowych były prowadzone na Politechnice Warszawskiej przez profesora Wacława Szymanowskiego, który wykorzystał swój wcześniejszy dorobek naukowy zgromadzony w okresie wojny w laboratoriach amerykańskich. W grupie współpracowników prof. Szymanowskiego, obok dr Witolda Kruczka, brał udział również autor niniejszego artykułu. Uzyskane osiągnięcia zostały udokumentowane konstrukcją prototypowej mozaikowej drukarki elektrograficznej dla WZE „Elwro", zrealizowanej w Instytucie Fizyki Politechniki Warszawskiej. Prace wstrzymano po zakupieniu przez Polskę licencji angielskiej na mechaniczne drukarki wierszowe „on-the-fly" z wirującym bębnem czcionkowym (produkowane nadal przez Zakłady w Błoniu).

[3]) Papier ten, łatwy w produkcji, kosztuje około trzech razy drożej od papieru zwykłego.

- kontrola jaskrawości.

Cechy podsystemu pamięciowego:

- stałość bądź zmienność rekordów (zapisów na nośniku pamięciowym),
- maksymalna długość rekordów,
- centrowanie nagłówków (tytułów),
- automatyczne podkreślanie,
- automatyczna identyfikacja,
- numerowanie stron,
- numerowanie akapitów,
- wyrównywanie dziesiętne (decimal alignment),
- detekcja korekty błędów,
- podpowiedzi ekranowe,
- prezentacja poleceń formatu,
- jednooperacyjne naprowadzenie na początek strony, dokumentu, tekstu,
- automatyczne wstawki znaku powrotu karetki w celu przeciwdziałania przełamywaniu słów,
- wymazywanie słów,
- możliwość adresowania partii tekstu,
- wymuszony powrót karetki ze względu na „szpaltowanie",
- wymuszone „cytowanie" (ang. required hyphen).

Możliwości redakcyjne:

- czy tekst źródłowy (pierwotny) zostaje zachowany,
- czy tylko wybrane części tekstu na dysku mogą być zabezpieczone przed zmianami,
- typ parametru wyszukującego (np. napis, numer wiersza, znacznik, podkreślenia, korekta),
- automatyczne przenumerowanie stron po zmianie formatu,
- lista operacji (np. wstaw, usuń, zmień, przesuń, powtórz), na znakach, słowach, wierszach, kolumnach, zdaniach, akapitach, stronach,
- automatyczna zmiana formatu według długości strony, długości wiersza (tzw. łamanie tekstu),
- automatyczne usuwanie porzuconych fragmentów tekstu (ang. widow prevention) w celu oczyszczenia pamięci,
- identyfikacja poprawek na ekranie oraz drukarce,
- korekta okien (na przykład na ilustracje),
- indeksowanie wsteczne (ang. reverse index),
- skorowidze i słowniki terminów.

Cechy podsystemu drukującego:

- poziom hałasu drukarek,
- typ drukarki,
- możliwość uruchamiania przy pisaniu na klawiaturze nowego tekstu,
- opcja zmiany kroju pisma, np. druk wytłuszczony, kursywa, itd.,
- możliwość wydruku na taśmie bez końca (ang. continuous stationery),
- możliwość wydruku na stronach jednostkowych, na przykład na formularzach jednostkowych,
- możliwość zmiany fizycznego formatu wydruku,
- możliwość nadruków na kopertach,
- możliwość wydruku na sztywnych kartach kartotekowych.
- możliwość drukowania etykiet, na przykład druk etykiet adresowych,
- możliwość automatycznego podawania i odbierania papieru,
- automatyczne podajniki formularzy jednostkowych,
- wielodostępność drukarek,
- szybkość drukowania (efektywna, z uwzględnieniem podawania papieru),
- typy taśmy barwiącej (brak, zwykła, karbonizowana, jednobarwna, wielobarwna, z wycieraniem (wybielająca)),
- maksymalna szerokość papieru,
- minimalny format strony,
- kopie węglowe,
- wybór kształtu i wielkości czcionek,
- fizyczna kontrola marginesów,
- indeksowanie fizycznego położenia głowisy drukującej względem papieru, na przykład orientacja na lewy skrajny róg formularza.

Możliwości dodatkowe:

- czy jest podsystem komunikacji teletransmisyjnej (jeśli tak, to jaki jest typ protokołu),
- praca w trybie arytmetycznym lub w innym obliczeniowym,
- podsystem sterujący wydrukowane dokumenty,
- wejście automatyczne z optycznego czytnika znaków (OCR); wyjście na OCR,
- podsystem taśmy perforowanej lub kart perforowanych,
- czy komunikacja da się realizować podczas przetwarzania,
- globalne wyszukiwanie i wymiana danych,
- sprzęg do pracy z fotoskładem (współpraca z maszynami poligraficznymi),
- dystrybucja fizyczna tekstów (przesyłki pocztowe, poczta pneumatyczna, komunikacja elektroniczna),
- sprzęganie akustyczne przez linie telefoniczne.

LITERATURA

[1] Bieńkowski K.: Dialogowe hierarchiczne systemy graficzne. Raporty Badawcze Instytutu Informatyki PW, nr 43, 1982

[2] Bieńkowski K., Kasprzak W.: Komputery dostosowywalne. Elektronika, nr 5, str. 29—33, 1983

[3] Bieńkowski K., Kasprzak W.: O dostosowywalności komputerów. Prace Naukowe Politechniki Warszawskiej, Elektronika, z. 64, str. 7—31, 1985

[4] Cherry P. D.: Word Processing System Design for High Throughput. Computer Design, pp. 95—99, August 1980

[5] Macioł A., Stawowy A.: Środki do przetwarzania tekstów — historia i dzień bieżący, Informatyka, nr 5, str. 20—24, 1986.

Nowe książki

Wydawnictwa Naukowo-Techniczne

Wacław Iszkowski — Nauka programowania w języku BASIC dla początkujących. Warszawa 1987, 100 000 egz., 256 str., cena 450 zł

W książce omówiono podstawowe elementy języka BASIC dla mikrokomputerów. Zwrócono szczególną uwagę na nauczenie prawidłowego programowania. Tekst zilustrowano przykładami i rysunkami, przygotowanymi za pomocą komputera. Przedstawiono również problemy do samodzielnego rozwiązania. Książka jest przeznaczona dla szerokiego kręgu czytelników, mających dostęp do popularnych komputerów.

Jerzy Grabowski, Stanisław Koślacz — Podstawy i praktyka programowania mikroprocesorów. Warszawa 1987 r., wyd. 2, 20 000 egz., 460 str., cena 500 zł

W książce przedstawiono praktyczne zagadnienia programowania mikrokomputerów, strukturę mikroprocesorów i otoczenia w jakim pracują. Za podstawę rozważań przyjęto mikroprocesor Intel-8080. Podano możliwości programowe mikroprocesorów ośmiobitowych Z-80 i Motorola 6800 oraz szesnastobitowych Motorola 68000 i Intel 8086. Omówiono różne techniki programowania innych systemów komputerowych. Materiał zilustrowano przykładami różnych rozwiązań.
Książka jest przeznaczona dla szerokiego kręgu czytelników interesujących się mikrokomputerami.

Wydawnictwa MIT PRESS

L. Sterling, E. Shapiro: The Art of Prolog — Advanced Programming Techniques. 1986.

Wprowadzenie do Prologu i języków programowania logicznego. Przedstawiono nowoczesne techniki programowania w Prologu i omówiono szczegółowo takie zastosowania jak: gry, system ekspertowy do oceny kredytów, program obliczeń symbolicznych oraz przykładowy kompilator.

E. Shapiro (red.): Concurrent Prolog. 1987.

Zbiór artykułów poświęconych metodom programowania, zastosowaniom i technikom realizacji języka Concurrent Prolog, umożliwiającego współbieżne programowanie i równoległe wykonywanie programów.

ZDZISŁAW PŁOSKI (oprac.)
Instytut Informatyki
Uniwersytet Wrocławski

Język awk — przetwarzanie wzorców

Awk jest językiem przeznaczonym do wyszukiwania informacji i operowania danymi. Od wprowadzenia (w 1979 roku) stał się w USA popularny nawet wśród ludzi nie przygotowanych do programowania. Celem tego artykułu, opracowanego na podstawie [1], jest przybliżenie specyfiki języka awk, którego koncepcje w dziedzinie operowania wzorcami tekstowymi są bez wątpienia interesujące i warte naśladowania. Artykuł jest pobieżnym przeglądem właściwości języka dokonanym wyłącznie na podstawie źródłowego materiału piśmienniczego i bez doświadczeń praktycznych.

STRUKTURA PROGRAMU

Podstawową operacją systemu awk jest przeglądanie kolejnych wierszy tekstu w poszukiwaniu takich, które odpowiadają jakiemuś spośród określonych przez użytkownika wzorców, czyli spełniają któryś z zadanych warunków. Znajduje to odzwierciedlenie w postaci programu. Programem w języku awk jest ciąg instrukcji o dwudzielnej budowie; każda z instrukcji składa się z części wyrażającej wzorzec i części określającej działanie:

wzorzec { działanie }
wzorzec { działanie }
. . .

Przykładowo, następujący wiersz tekstu

```
$1 == "adres" { print $2, $3 }          (1)
```

jest programem wyrażonym za pomocą jednej instrukcji (przykłady będące kompletnymi programami w języku awk będą zaznaczane kolejnymi numerami). W wyniku wykonania tego programu nastąpi wydrukowanie z pliku wejściowego drugich i trzecich pól z wszystkich tych wierszy, które zawierają w pierwszym polu napis adres (definicję pola w języku awk podano dalej).

Każdą z dwu części instrukcji awk można opuścić. Pozostawienie samego wzorca, jak w programie:

```
$1 == "nazwa"                           (2)
```

powoduje wydrukowanie każdego wiersza zgodnego z tym wzorcem. Jeśli nie określi się żadnego wzorca, lecz tylko jakieś działanie, np.:

```
{ print $1, $2 }                        (3)
```

to zostanie ono odniesione do każdego wiersza wejściowego. Właśnie dlatego, że zarówno wzorzec, jak i określenie działania mogą być w instrukcji pominięte, część określającą działanie ujmuje się — dla odróżnienia — w nawiasy klamrowe.

W przypadku wykazu wzorców (i działań), każdy wiersz wejściowy jest porównywany kolejno ze wszystkimi wzorcami wykazu, a działanie jest określone według wyniku porównania. Program:

```
$1 == "adres"  { print $2, $3 }         (4)
$1 == "nazwa"
               { print $1, $2 }
```

otrzymany przez połączenie w jeden program — instrukcji (1), (2) i (3) — można więc zinterpretować następująco „dla każdego wiersza pliku, jeśli pierwsze pole równa się adres, to drukuj pole drugie i trzecie; jeśli pierwsze pole równa się nazwa, to drukuj cały wiersz; niezależnie od przypadku drukuj pole pierwsze i drugie".

SPOSÓB UŻYCIA

Najprostsze, zwykle polegające na wyszukiwaniu informacji, zastosowania języka awk można wyrazić w postaci bardzo zwięzłych programów, często będących pojedynczymi instrukcjami. Z tego powodu systemu awk można używać dwojako:

- konwersacyjnie, umieszczając krótkie programy wprost w poleceniu uruchamiającym awk, na przykład:

```
awk 'print $1, $2' plik1 plik2 ...
```

(programy ujmuje się wówczas w apostrofy, aby zapewnić przekazanie ich tekstu interpreterowi awk w stanie nienaruszonym przez powłokę systemu operacyjnego);

- z rozdzieleniem faz przygotowania i uruchomienia programu, tj. ze wskazaniem w poleceniu awk pliku z programem do wykonania:

```
awk -f programtaki dane1 dane2 ...
```

POLA

Awk rozbija czytany wiersz tekstu na ciąg pól. Jeśli nie określono inaczej, to za pole przyjmuje się ciąg znaków nie zawierający odstępów ani znaków tabulacji.

Awk zawiera 11 zmiennych systemowych, których wartości mogą być modyfikowane w programie. Jedną z nich jest zmienna **FS**, zawierająca informację o aktualnym separatorze pól. Pola mają przyporządkowane numery, poczynając od 1. Odniesienie do pola jest oznaczane przez poprzedzenie określenia numeru pola znakiem $, np. $1, $4, $i. Cały wiersz oznacza się przez $0. Numer pola może być obliczony jako wartość wyrażenia. Na przykład, $ (NF — 1) oznacza pole przedostatnie (standardowo zmienna **NF** zawiera liczbę pól danego wiersza). Odwołanie do pola nie istniejącego, np. $ (NF + 1), daje wartość pustą.

W typowej implementacji liczbę pól w wierszu ogranicza się do 100, a długość pola — do 1024 znaków (przy maksymalnej długości wiersza ustalonej na 2500 znaków).

Oto przykład pliku (nazywanego dalej **stany**), zawierającego typową dla awk mieszankę danych — słów i liczb. Każdy wiersz pliku zawiera: nazwę stanu, jego powierzchnię w km², zaludnienie (w mln osób, wg stanu na rok 1976) oraz wyjaśnienie znaczenia nazwy (źródłosłowy podano za [2]).

Arizona	**295**	**2.2**	**małe źródło**
Oregon	**251**	**2.3**	**ludzie grubouchy**
Minnesota		**218**	**4.0 niebieska woda**
Idaho	**216**	**0.831**	**klejnot gór**
Nebraska		**200**	**1.6 płytka woda**
Oklahoma		**181**	**2.8 czerwoni ludzie**
Missouri		**180**	**4.8 mętna rzeka**
Illinois		**146**	**11.2 prawdziwi ludzie**
Ohio	**107**	**10.7**	**piękna rzeka**
Kentucky		**105**	**3.4 zielona trzcina**

Przy założeniu standardowych wartości separatora pól, każdy wiersz powyższego pliku składa się z pięciu pól. Pierwsze cztery pola oddzielają od siebie znaki tabulacji, ostatnie pole w każdym wierszu oddziela od reszty odstęp.

Wartości pól można określać w programie na nowo, za pomocą przypisania. Przykładowo, program:

```
$4 == niebieska { $4 = obloczna }       (5)
```

zmieni w pliku **stany** wartość czwartego pola w trzecim wierszu na **obloczna**.

DRUKOWANIE

Program:

```
awk '{ print }' stany
```

wpisany w polecenie uruchamiające awk, spowoduje wydrukowanie całego (gdyż nie występuje wzorzec wyróżniający wiersze) pliku **stany**. Za pomocą instrukcji **print** można też drukować wybrane części wierszy pliku, np. program:

```
{ print $1, $3 }                                    (6)
```

wydrukuje tylko pola pierwsze i trzecie z pliku **stany** (ten i następne przykłady odnoszą się stale do pliku **stany**). Wiersze w pliku wynikowym przyjmą postać:

```
Arizona 2.2
Oregon 2.3
Minnesota 4.0
...
```

Jednostki oddzielone w instrukcji **print** przecinkami są rozdzielane przy wyprowadzaniu przez **separator pól wynikowych** (zastępczo — odstęp), a każdy wiersz wyjściowy jest zakończony **separatorem rekordów wynikowych** (standardowo — znakiem nowej linii). Instrukcja **print** służy do niewyszukanego, doraźnego drukowania. Staranne opracowanie formy wyprowadzanych tekstów umożliwia instrukcja:

printf format, wyrażenie$_1$, wyrażenie$_2$ ... wyrażenie$_n$

bardzo podobna do instrukcji **printf** w języku C (wiele elementów składni awk wywodzi się z języka C). Za pomocą tej instrukcji powyższe dane można wyprowadzić w postaci prawo- lub lewostronnie wyrównanych kolumn. Przykładowo, wynikiem działania programu:

```
{ printf "%10s %6.1f\n", $1, $3 }                   (7)
```

będzie wydruk:

```
Arizona      2.2
Oregon       2.3
Minnesota    4.0
```

W przeciwieństwie do **print**, przy drukowaniu za pomocą **printf** nie dodaje się automatycznie żadnych separatorów; formę drukowanych wyników należy określać jawnie w pierwszym parametrze instrukcji.

Awk uaktualnia automatycznie stan swoich systemowych zmiennych **NR** i **NF**. Pierwsza z nich przechowuje numer bieżącego wiersza wejściowego, druga — liczbę pól. Na przykład, program:

```
{ print NR, NF, $0 }                                (8)
```

drukuje każdy wiersz wejściowy z dodaniem (na początku wiersza) jego numeru i liczby pól, z których się składa.

NAJPROSTSZE WZORCE I WYRAŻENIA ARYTMETYCZNE

Wzorce w języku awk służą wyszukiwaniu informacji. Wiersze „przystające" do wzorców są poddawane obróbce przez stowarzyszone z danym wzorcem działanie. Aczkolwiek w ogólnym przypadku programowanie wzorców może być zadaniem złożonym (proporcjonalnie do zawiłości kontekstu, który ma być określony), w prostych zastosowaniach wzorce są łatwe do interpretacji — nawet intuicyjnej. Oto przykłady.

Aby wydrukować z pliku **stany** te wiersze, w których opisano stany mające w źródłosłowie nazwy wyraz **ludzie**, wystarczy napisać:

```
/ludzie/                                            (9)
```

Powyższy program wydrukuje każdy wiersz, który zawiera dowolnie położony — także jako fragment dłuższego słowa — co najmniej jeden napis **ludzie**. Użyty wzorzec stanowi przykład najprostszej postaci tzw. **wyrażenia regularnego** (tu: ciąg znaków ograniczony obustronnie przez /). Inny wzorzec —

```
$4 == "ludzie"                                      (10)
```

ogranicza zakres poszukiwania napisu **ludzie** do czwartego pola każdego wiersza (przez napis jest rozumiany ciąg znaków wewnątrz cudzysłowów). W rozważanym przypadku zostanie wydrukowane tylko:

```
Oregon 251 2.3 ludzie grubouchy
```

Oprócz operatora równości, są określone wszystkie inne operatory porównania (>, > = itd.) — zarówno dla tekstów, jak i dla liczb. Awk według kontekstu interpretuje napis jako wartość liczbową lub tekstową. W ten sposób warunek-program:

```
$2 >= 200                                           (11)
```

spowoduje wyselekcjonowanie z pliku informacji tylko o tych stanach, w których opisach drugie pola zawierają liczbę nie mniejszą niż 200:

```
Minnesota   218   4.0   niebieska woda
Nebraska    200   1.6   płytka woda
```

Dwa specjalne wzorce „przystają" do początku i końca pliku. Są to: **BEGIN** i **END**. Umożliwiają one umiejscowienie pewnych działań przed pobraniem pierwszego wiersza z wejścia i po przetworzeniu ostatniego. Na przykład, program wyprowadzający dane o stanach zawierających w nazwie pojęcia **rzeka** lub **woda** i obliczający ich średnią gęstość zaludnienia może mieć postać:

```
BEGIN { print "Stany z rzeką lub wodą w nazwie:" }  (12)
$5 == "rzeka" || $5 == "woda" { print "  ", $1
                                suma = suma + $3
                                ls += 1 }
END { printf "%0.4f", „Zaludnienie średnie:", suma / ls }
```

Przykład ten jest też ilustracją sposobu zaprogramowania prostych działań numerycznych w języku awk. Łatwo rozpoznać tu instrukcje przypisania; ich składnia i interpretacja nie odbiega od rozwiązań przyjętych w języku C. Każda zmienna może przyjmować wartość liczoną lub napisową. Przy pierwszych odwołaniach awk przypisuje automatycznie każdej zmiennej wartości 0 lub "" (napis pusty) — w zależności od sposobu potraktowania zmiennej. Wzorzec określający żądane rekordy jest tutaj kombinacją dwu prostszych warunków — operator || oznacza alternatywę wzorców (podobnie && — koniunkcję, a ! — negację). Program wydrukuje następujące wyniki:

```
Stany z rzeką lub wodą w nazwie:
   Minnesota
   Nebraska
   Missouri
   Ohio
Zaludnienie średnie: 4.4
```

RELACJE I WYRAŻENIA REGULARNE

W operacjach porównania napisów i liczb obowiązuje zasada: jeśli oba argumenty są numeryczne, to wykonuje się porównanie arytmetyczne (awk prowadzi obliczenia zmiennoprzecinkowe), w przeciwnym razie argumenty relacji są porównywane jako napisy.
Program:

```
$1 >= "N"                                           (13)
```

wyprowadzi wszystkie wiersze, które rozpoczynają się od liter alfabetycznie późniejszych lub równych **N**:

```
Oregon      251   2.3   ludzie grubouchy
Nebraska        200   1.6   płytka woda
Oklahoma        181   2.8   czerwoni ludzie
Ohio        107   10.7  piękna rzeka
```

Dwa specjalne operatory: ~ i !~ służą do porównywania napisów z wartościami wyrażeń regularnych. Oznaczają one odpowiednio tekstową tożsamość wartości wyrażeń albo jej niespełnianie. Przy użyciu operatora ~ program (10) można zapisać jako:

```
$4 ~ /ludzie/                                       (14)
```

W wyrażeniach regularnych znaki: ukośnik, daszek, $, [], *, +, ?, (), : . symbolizują określone konteksty bądź pełnią funkcje porządkujące. Tak więc, metaznakowi daszek odpowiada początek przyrównywania napisu, $ — przystaje do końca napisu, a . dopasowuje pojedynczy znak. W ten sposób program:

```
^. $/                                               (15)
```

wyszuka z pliku wszystkie te wiersze, które mają tylko po jednym znaku. Nawiasy [ ] grupują alternatywne znaki. Wzorzec **/[AIK]/** zostanie dopasowany do każdego wiersza zawierającego literę **A** albo **I**, albo **K**. W odniesieniu

do liter i cyfr znak minus wewnątrz nawiasów [ ] znamionuje przedział; na przykład, **/[a-zA-Z]/** dopasowuje każdą literę. Jeśli pierwszym znakiem po [jest znak daszek, to za zbiór dopasowywanych znaków bierze się dopełnienie (w zbiorze ASCII); na przykład, **/[^a-zA-Z]/** odpowiada dowolnemu znakowi różnemu od litery.

Opatrzenie wykazu w nawiasach [ ] znakiem + lub * powoduje dopasowanie dowolnego (w przypadku + niepustego) spójnego ciągu podanych w wykazie (zbiorze) znaków. Program:

```
$3 ~ /^0|.[0—9]+$/                                    (16)
```

wyszuka w pliku **stany** wiersz:

Idaho 216 0.831 klejnot gór

jedyny, w którym trzecie pole rozpoczyna się od zera, po którym następuje kropka (ukośnik, symbolizowany przez kreskę pionową, chroni przed potraktowaniem kropki jako metaznaku), a następnie ciąg cyfr i nic więcej ($). W przykładzie ukazano, jak wyrażenia regularne mogą służyć do badania poprawności danych, co również należy do zastosowań języka awk.

Nawiasy ( ) wraz z operatorem | służą w wyrażeniach regularnych do budowania alternatyw napisów. Przykładowo, warunek z programu (12) można wyrazić w postaci:

```
$5 ~ /^(rzeka|woda)$/
```

Oprócz metaznaków awk interpretuje w wyrażeniach regularnych symbole redagowania graficznego — takie jak w języku C. Na przykład, **\n** oznacza nową linię, **\t** — tabulację itp.

Niezależnie od tego, czy wyrażenie na prawo od operatora ~ lub !~ jest podane w ogranicznikach / /, za pomocą napisu czy przez nazwę zmiennej, awk, interpretuje ten kontekst jako wyrażenie regularne. Wobec tego, program:

```
BEGIN { cyfry = "^[0—9]+$" }                          (17)
$2 !~ cyfry
```

wyprowadzi wszystkie takie wiersze, których drugie pola zawierają znaki różne od cyfr dziesiętnych.

PRZEDZIAŁY WZORCÓW

Dwa wzorce oddzielone przecinkiem:

$wzorzec_1$, $wzorzec_2$ {...}

tworzą przedział, co oznacza, że odpowiednie działanie będzie odnosić się do wszystkich wierszy, poczynając od wiersza dopasowanego do wzorca pierwszego, a kończąc na wierszu odpowiadającym wzorcowi drugiemu. Na przykład program:

```
/Nebraska/, /Illinois/                                (18)
```

wydrukuje wiersze:

Nebraska	**200**	**1.6**	**płytka woda**
Oklahoma	**181**	**2.8**	**czerwoni ludzie**
Missouri	**180**	**4.8**	**mętna rzeka**
Illinois	**146**	**11.2**	**prawdziwi ludzie**

NAPISY I TABLICE

Napis określa się jawnie jako ograniczony cudzysłowami ciąg znaków. Prócz tego, wartości napisowe mogą być reprezentowane przez pola, zmienne, wyniki działania funkcji napisowych, elementy tablic i inne wyrażenia.

Dwa wyrażenia reprezentujące napisy, napisane obok siebie, dają w wyniku połączenie tekstowe (konkatenację) swych wartości. Nie używa się w awk żadnego specjalnego symbolu w tym celu. Przykładowo, program:

```
{ print NR ":" $0 }                                   (19)
```

powoduje przekopiowanie każdego wiersza wejściowego z dodaniem numeracji oddzielonej od wejściowej zawartości wiersza dwukropkiem (bez odstępów).

Awk pozwala używać jednowymiarowych tablic (zarówno tablic, jak i zmiennych nie deklaruje się). Indeksami elementów tablic mogą być liczby lub napisy. Oto zarys programu, który przed podjęciem innych działań wczytuje cały plik do pamięci:

```
        { x [NR] = $0 }
END { ... pozostałe działania ... }
```

Do poszczególnych wierszy pliku wejściowego można się będzie w tym programie odwoływać jako do elementów tablicy x. Natomiast instrukcja:

```
        { stan [$1] = $4 $5 }
END { pozostałe działania }
```

zakłada w programie tablicę **stan** indeksowaną nazwami stanów. Dalej można się odwoływać do tej tablicy na przykład tak:

```
print { stan ["Arizona"] "i" stan ["Kentucky"] |
        "— tyle znaczą w indiańskich narzeczach nazwy \n" |
        "Arizona i Kentucky." }
```

Znak | pozwala kontynuować długą instrukcję języka awk w następnym wierszu (w tym artykule zastępuje on ukośnik).

Dowolny element tablicy może zostać z niej usunięty za pomocą instrukcji delete o składni:
delete nazwa_tablicy [indeks]

Dowolny napis można rozbić na pola będące elementami tablicy za pomocą standardowej funkcji **split**. Na przykład wywołanie:

```
split { "godziny:minuty:sekundy", czas, ":" }
```

spowoduje zapamiętanie fragmentów napisu **"godziny:minuty:sekundy"** w tablicy **czas: godziny — w czas[1], minuty — w czas[2] i sekundy — w czas[3]**. Trzeci parametr funkcji **split** określa separator podziału. W razie braku tego parametru korzysta się z aktualnej wartości zmiennej systemowej FS.

Napisy reprezentujące indeksy można tworzyć z liczb i separatorów, na przykład **tablica [i "," j]**. Uzyskuje się w ten sposób możliwość indeksowania wielowymiarowego.

Split jest jedną z siedmiu standardowych funkcji napisowych awk. Pozostałe wykonują inne działania: funkcje **sub** i **gsub** zamieniają wybrane fragmenty napisów lub wierszy, przy czym liczba dokonanych zmian jest wynikiem tych funkcji; **index** znajduje miejsca występowania podnapisów; **length** określa długość wiersza lub napisu; **substr** wyodrębnia fragmenty napisów, a **sprintf** formatuje wartości wyrażeń napisowych. Funkcja **sprintf** nie powoduje przy tym drukowania (!) — sformatowane (na podobieństwo instrukcji **printf**) napisy należy umieścić, na przykład, w zmiennej. A oto przykład programu wyszukującego najdłuższą nazwę stanu:

```
length ($1) > max  { max = length ($1) ; nazwa = $1 }  (20)
END                { print nazwa }
```

W wyniku zostanie wydrukowana nazwa:

Minnesota

INSTRUKCJE STERUJĄCE

Awk zawiera instrukcje **if-else**, **while** i **for** (oraz możliwość grupowania instrukcji za pomocą nawiasów klamrowych), zbudowane tak jak w języku C. Oto przykłady ich użycia.
Program:

```
{ if maxlud < $3 {                                    (21)
        maxlud = $3
        stan = $1
  }
}
END     { print stan, maxlud }
```

wydrukuje w wyniku:

Illinois 11.2

Wydrukowanie pliku wejściowego po jednym polu w wierszu można zaprogramować następująco:

```
{ i = 1                                                    (22)
  while (i <= NF) {
        print $i
        i++
  }
}
```

lub równoważnie:

```
{ for (i = 1; i <= NF; i++) print $i }                     (23)
```

Dziedzictwo cech języka C jest tu widoczne wyraźnie.

Analogiczna jak w języku C, jest rola instrukcji **break** i **continue** (przerwanie wykonywania pętli, natychmiastowe przejście do ponowienia pętli). Instrukcja **next** powoduje natychmiastowe pobranie następnego wiersza i rozpoczęcie czynności przyrównywania do nowego wiersza wzorców (z ewentualnym podejmowaniem dalszych działań) od początku. Instrukcja **exit** wstrzymuje czytanie z wejścia i powoduje wykonanie działań związanych z wzorcem **END**, jeśli takie zostały określone.

W przypadku działań na tablicach, warunkiem w instrukcji **if** lub zmienną sterującą w instrukcji **for** może być wyrażenie postaci:

(zmienna **in** tablica)

Elementy tablicy dla instrukcji:

for (zmienna **in** tablica) instrukcja

są generowane w losowej kolejności (!).

OPISYWANIE FUNKCJI

Oprócz standardowych funkcji napisowych, awk zawiera komplet funkcji arytmetycznych (**sin, cos, atan2, sqrt, exp, log, int, rand, srand**). Ponadto, użytkownik może definiować własne funkcje. Termin **funkcja** jest używany w awk tak jak w języku C i obejmuje zarówno pojęcie funkcji, jak i procedury w rozumieniu, na przykład, języka Pascal. Opisy funkcji w postaci:

```
func nazwa (lista_argumentów) {
        instrukcje
}
```

mogą wystąpić w miejscu dowolnej instrukcji. Tablice w argumentach funkcji awk są przekazywane przez adres (jak w języku C), a argumenty skalarne — w postaci obliczonych wartości. W treści funkcji poza parametrami formalnymi, które są lokalne, wszystkie zmienne są globalne (!). Wolno jednak użyć dodatkowych parametrów formalnych w celu uzyskania efektu zmiennych lokalnych w treści funkcji. Funkcja awk ma określoną wartość tylko wówczas, gdy wykonywanie jej treści zakończyła instrukcja **return wyrażenie** (z niepustym parametrem). W przeciwnym razie wartość funkcji jest nieokreślona ("procedura").

W opisach funkcji — i gdziekolwiek indziej — mogą występować komentarze. Za **komentarz** przyjmuje się części wierszy leżące na prawo od znaku # (nie będącego elementem napisu itp.).

Oto przykład programu, w którym zdefiniowano i testuje się funkcję **silnia:**

```
func silnia (n) {
        if (n <= 1)
                return 1
        else
          return n * silnia (n—1)
}
{ print $1 "! wynosi " silnia ($1) }
```

W wywołaniach nie można oddzielać odstępami nazw funkcji od nawiasów "(", gdyż awk potraktowałby te elementy jako argumenty konkatenacji.

OPEROWANIE DANYMI NA WEJŚCIU I WYJŚCIU

Programy napisane w języku awk mogą kierować wyniki do wielu plików (maksimum 15 otwartych jednocześnie). Odpowiednich przeadresowań dokonuje się w instrukcjach print (printf) za pomocą wyrażenia >nazwa_pliku. Na przykład, program:

```
{ print $1, $2, $3 > "daneadm"                             (25)
  print $1, $4, $5 > "daneetnogr" }
```

dokonuje rozdzielenia pliku **stany** na dwa pliki, nazwane tu **daneadm** i **daneetnogr.**

Można także programować w awk dalsze przetwarzanie wyników działania programu przez spajanie (ang. piping, tworzenie potoków), wyjścia awk z wejściami innych programów, na przykład:

```
{ print $1 " to znaczy " $4 $5 | "sort" }                  (26)
```

Tutaj **sort** jest nazwą programu, który przechwytuje wiersze wynikowe awk, porządkując plik leksykograficznie.

Przez zmianę standardowej wartości separatora rekordów wejściowych (**RS**) i separatora pól (**FS**), w rodzaju:

```
BEGIN { RS = ""; FS = "|n" }
```

można przeprowadzać operacje na rekordach złożonych z wielu wierszy. Ograniczeniem jest maksymalna długość rekordu. Gromadzenie rekordów złożonych z wielu wierszy można zorganizować przez wczytywanie poszczególnych wierszy za pomocą funkcji **getline** i pamiętanie ich w tablicy, na przykład:

```
^START/ {
        w[nw=1] = $0
          w [++ nw] = $0
        while (getline && $0 !~ /^STOP/)
        # teraz przetwarzanie danych w w[1]...w[nw]
        ...
}
```

Funkcja **getline** rozbija czytane wiersze w sposób analogiczny do automatycznego wejścia awk (wydzielanie pól, określanie zmiennych **NF, NR** itd.). W przykładzie jest widoczna inicjalizacja indeksu tablicy **w;** nie można poprzestać tu na automatycznym zapoczątkowaniu zmiennej **nw** wartością 0, gdyż indeksy tablic muszą się rozpoczynać od 1. Korzysta się tu również z właściwości zaprzestania obliczania (od lewej do prawej) koniunkcji &&, gdy tylko jest to możliwe, przy czym funkcja getline dostarcza wartość 1 (prawda) tylko wówczas, gdy wiersz został przeczytany poprawnie.

Równoważne działanie bez użycia funkcji **getline** może być zaprogramowane następująco:

```
/^START/ && nw==0 { w [nw=1] = $0 }
/^STOP/             { w [++nw] = $0 }
nw > 1              { # teraz przetwarzanie danych w
                    tablicy w
                    ...
                    nw = 0
                    }
```

Fnkcja **getline** może czytać dane z różnych plików (**getline** < plik), może również przenosić wiersze do zmiennych bez wyodrębniania pól (**getline** zmienna < plik).

Systemowa tablica **ARGV** przechowuje parametry z polecenia awk, za pomocą którego uruchamia się program w języku awk. Zmienna **ARGC** zawiera liczbę tych parametrów.

* * *

Jak stwierdzają autorzy raportu o języku awk, siłą tego języka jest możliwość organizowania w nim współpracy z innymi programami. Standardowa funkcja **system**(polecenie_dla_systemu_operacyjnego) pozwala uruchamiać z poziomu programu nadzorowanego przez interpreter awk inne programy.

Język awk jest przydatny zwłaszcza do sporządzania raportów, zestawień, wyciągów i tym podobnych dokumentów. Często stosuje się przy tym technikę wieloetapowego przetwarzania plików za pomocą programów napisanych w języku awk. Cechą takich programów jest ich zwartość — zwykle są krótsze niż ich odpowiedniki w języku C lub Pascalu.

dokończenie na str. 31

MIECZYSŁAW KŁOPOTEK
Instytut Podstaw Informatyki
Polskiej Akademii Nauk
Warszawa

Maszynowe tłumaczenie tekstów

Automatyczne tłumaczenie (AT) tekstów w języku naturalnym należy do najtrudniejszych zadań przetwarzania języka naturalnego. Chociaż pierwsze wysiłki w tej dziedzinie podjęto już w 1949 r. [25], jednak do dziś nie opracowano jeszcze systemu generującego tłumaczenia o satysfakcjonującej jakości. Mimo rozczarowań po początkowym okresie oczekiwań na szybki sukces, prace nigdy nie ustały całkowicie. Ciągłe zainteresowanie automatycznym tłumaczeniem było podtrzymywane, m.in.:

- zapotrzebowaniem na bieżące śledzenie obfitej literatury obcojęzycznej przy słabym opanowaniu języków przez zainteresowanych i przy niedostatku tłumaczy,
- koniecznością tłumaczenia oficjalnych dokumentów we wspólnotach wielojęzycznych,
- celową polityką państw wielonarodowościowych, zmierzającą do podtrzymywania kultury wielojęzycznej.

Mimo ogromnych nakładów finansowych na początku lat sześćdziesiątych (w 1965 roku w USA — 20 mln dolarów) i widocznych postępów, systemy AT nie spełniały pokładanych w nich nadziei. Sięgnięto więc po inne narzędzia mające zwiększyć wydajność pracy tłumaczy:

- interakcyjne systemy tłumaczenia, w których trudne decyzje podejmuje człowiek, a komputer dokonuje rutynowych czynności analizy tekstu źródłowego, przeglądu słowników i syntezy tekstu wynikowego,
- systemy słownikowe (których zadaniem jest zapewnienie szybkiego dostępu do najbardziej aktualnego słownictwa specjalistycznego) i inne pomoce dla tłumacza (edytory tekstu z wbudowanymi procedurami poprawnego sylabizowania, sprawdzania poprawności ortograficznej, kopiowania tabel, schematów i rysunków z tekstu źródłowego do docelowego itp.).

SYSTEMY SŁOWNIKOWE

Reakcją na coraz większe zapotrzebowanie na aktualne słowniki terminów naukowych, technicznych i ekonomicznych, zwłaszcza w dużych jednostkach rządowych i przemysłowych, było stworzenie systemów automatycznych słowników (AS). Znane są bardzo dobrze trudności, jakie napotyka tłumacz: szybko zmieniające się słownictwo, pojawianie się nowych pojęć, nowych technologii i wyrobów, często niewystarczająca standaryzacja terminologii, mnogość źródeł informacji o różnej jakości i wiarygodności. Stwierdzono, że tłumacz spędza do 60% swego czasu przeglądając słowniki [17]. Jeśli zważy się, że w dużych jednostkach organizacyjnych różne osoby poszukują tych samych pojęć, to staje się oczywiste, jak wielkie oszczędności może przynieść system AS. Jedną z postaci takich systemów są tworzone przez centrale rządowe EWG oraz Kanady Terminologiczne Bazy Danych (TBD).

Sukces systemów AS w porównaniu z AT wynika stąd, że system AS nie musi rozróżniać form gramatycznych (przypadków, czasów, trybów itp.), wybierać znaczenia właściwego dla danego kontekstu, ani tym bardziej określać funkcji syntaktycznej i semantycznej wyrazu w zdaniu, gdyż pracę tę z łatwością wykona tłumacz.

Zadaniem wielojęzycznych TBD jest:

- dostarczanie informacji o pojedynczych słowach i frazach (definicje terminów, przykłady użycia i tłumaczenia),
- generowanie słowniczków odnoszących się do tłumaczonego tekstu,
- generowanie ogólnych aktualnych słowników i leksykonów.

Przykładami funkcjonujących systemów AS są:

- EURODICAUTOM — terminologiczna baza danych EWG, obejmująca wszystkie języki tej wspólnoty,
- TERMIUM — TBD Uniwersytetu w Montrealu, obejmująca terminy w językach angielskim i francuskim,
- LEXIS zachodnioniemieckiego Urzędu Języków, obejmujący pojęcia, zwroty naukowe i techniczne (w 1978 r. 1,2 mln pozycji),
- TEAM firmy Siemens, obejmujący w 1978 roku 700 000 terminów naukowo-technicznych w głównych językach europejskich (angielski, niemiecki oraz rosyjski, francuski, włoski, hiszpański, portugalski).

Dwie pierwsze TBD są ustawicznie poszerzane o informacje zebrane przez tłumaczy podczas ich normalnej pracy. Napotykając nowe pojęcie, tłumacz podaje przykładowy zwrot z tym terminem oraz jego równoważne tłumaczenie, źródło, kod dziedziny i bibliografię. Szersze omówienie systemów AS można znaleźć w pracy [18].

SYSTEMY INTERAKCYJNEGO TŁUMACZENIA

Współpraca człowieka z komputerem w systemach interakcyjnego tłumaczenia (IT) może przyjmować różne formy. Udział człowieka może być ograniczony do tych faz tłumaczenia, w których automatyzacja napotyka największe trudności (rozstrzyganie wieloznaczności syntaktycznych i semantycznych tekstu źródłowego), może też być rozszerzony o przygotowanie tekstu przed wprowadzeniem, np. upraszczanie wyrażeń, zaznaczanie relacji w zdaniu itp. lub dotyczyć wyboru zwrotów języka docelowego.

Możliwości systemów interakcyjnych nie zostały jeszcze całkowicie zbadane. Tworzone systemy mają charakter eksperymentalny, np.:

- MIND [13] (Rand Corporation), wyposażony w różnorodne narzędzia przetwarzania danych językowych z bazą wiedzy o „świecie rzeczywistym",
- system Provo z Birgham Young University (Utah, USA), tłumaczący tekst źródłowy równocześnie na kilka języków (z angielskiego na hiszpański, portugalski, francuski i niemiecki),
- CULT Uniwersytetu Chińskiego w Hong-Kongu tłumaczący teksty matematyczne i fizyczne z chińskiego na angielski, od 1973 r. stosowany stale do tłumaczenia wydań Acta Mathematica Sinaica.

Podejście zastosowane w systemie Provo wydaje się być najbardziej obiecujące, gdyż jednorazowy wysiłek człowieka na etapie analizy tekstu źródłowego w jednym języku owocuje automatycznym tłumaczeniem na kilka różnych języków.

SYSTEMY TŁUMACZENIA AUTOMATYCZNEGO

Systemy automatycznego tłumaczenia (AT) dostarczają na ogół tekstów o niezbyt dobrej jakości, dlatego wymagają weryfikacji przez człowieka. Ideałem byłoby, gdyby poprawki dotyczyły jedynie kwestii stylistycznych. Tak niestety nie jest, ale w ciągu swej czterdziestoletniej historii postęp teoretyczny w dziedzinie AT umożliwił wyeliminowanie wielu źródeł błędnych tłumaczeń. W rozwoju systemów AT wyróżnia się trzy generacje koncepcyjne:

- **systemy tłumaczenia bezpośredniego**, tłumaczące „słowo w słowo" z uwzględnieniem jedynie idiomów, i dokonujące niewielkich przestawień słów (brak analizy semantycznej);
- **systemy tłumaczenia pośredniego**, zakładające istnienie reprezentacji języka wspólnej dla wszystkich języków; tekst źródłowy tłumaczy się najpierw na sztuczną reprezentację pośrednią (dokonując szczegółowej analizy syntaktycznej i w miarę potrzeb powierzchniowej analizy se-

mantycznej), a dopiero ta reprezentacja pośrednia jest tłumaczona (syntetyzowana) na język docelowy;

● **systemy oparte na semantyce**, których twórcy uważają, że poprawne przetłumaczenie wymaga „rozumienia" tekstu na podstawie znaczeń poszczególnych słów i ogólnej wiedzy o świecie, a nie tylko znajomości struktury syntaktycznej.

Pierwsza generacja

Systemy tłumaczenia bezpośredniego — aczkolwiek prymitywne — są praktycznie jedynymi używanymi systemami AT.

Postęp w dziedzinie tych systemów można zobrazować porównaniem: w 1966 roku Doradcza Komisja ds. Automatycznego Przetwarzania Języka (ALPAC) Krajowego Funduszu Nauki USA opublikowała raport [19], stwierdzający, że AT jest wolniejsze, mniej dokładne i dwa razy droższe od tłumaczenia dokonywanego przez człowieka. W 1980 r. wprowadzono system PAHO [20], na którego hiszpańsko-angielskiej wersji dokonano eksperymentu z tekstem o długości 101 tys. słów. Tłumaczenie przez człowieka trwało 65 dni i kosztowało 8296 dolarów, a AT — łącznie z weryfikacją wykonaną przez człowieka 36 dni i 3217 dolarów.

Systemy pierwszej generacji traktują tekst jako sekwencję nie powiązanych zdań. Zdanie jest strumieniem słów i zwrotów idiomatycznych. Analiza syntaktyczna jest cząstkowa lub nie ma jej wcale. Podobnie jest z analizą semantyczną. Oprócz PAHO przykładami systemów pierwszej generacji są:
● MARK II [12], który tłumaczy z rosyjskiego na angielski, identyfikując tematy słów,
● SYSTRAN [22], który po 1970 roku zastąpił stosowany w amerykańskim lotnictwie system MARK II; w EWG istnieją wersje do tłumaczenia z angielskiego na francuski i odwrotnie oraz z angielskiego na włoski,
● LOGOS [3], tłumaczący z angielskiego na wietnamski.

Druga generacja

Wadą systemów pierwszej generacji jest ich ukierunkowanie na konkretny język źródłowy i docelowy. Tłumaczenie między 8 językami EWG wymagałoby więc 56 systemów AT. Tymczasem zauważono, że znaczna część procesu tłumaczenia jest specyfikowana dla jednego z języków i nie zależy od drugiego. Jeśli więc uda się znaleźć możliwie małą część systemu AT zależną od obu języków i możliwie duże części zależne tylko od jednego z języków, to nakład pracy niezbędny do realizacji tłumaczenia z dalszych języków (lub na dalsze języki) będzie zminimalizowany. Taką filozofię przyjęto właśnie dla systemu EUROTRA [14].

W zależności od tego, co jest częścią zależną od obu języków (źródłowego i docelowego), wyróżnia się dwa podejścia:
● interlingwistyczne, w którym zakłada się istnienie języka pośredniego jako części zależnej od różnych języków, a tłumaczenie przebiega w dwu fazach — tłumaczenie z języka źródłowego na język pośredni i tłumaczenie z języka pośredniego na język docelowy;
● transferowe, w którym przekształcenie struktury syntatycznej jednego języka na strukturę drugiego zależy od obu języków; tłumaczenie odbywa się w trzech fazach: przekształcenie tekstu źródłowego na strukturę syntaktyczną języka źródłowego, transfer (przekształcenie) struktury źródłowej na docelową oraz generowanie tekstu docelowego ze struktury syntatycznej docelowej.

Pierwsze podejście zastosowano w systemie CETA [23] (opracowanym na Uniwersytecie w Grenoble), o języku pośrednim „pivot", wolnym od ograniczeń morfologicznych i syntaktycznych języków naturalnych. Zasadniczo system tłumaczy z języka rosyjskiego na francuski, ale eksperymentowano też z językiem niemiecki i japońskim.

Podejście transferowe zastosowano w systemie EUROTRA, TAUM [15] i jego uproszczonej wersji TAUM-METEO [5] (Uniwersytet w Montrealu), GETA [2] (Uniwersytet w Grenoble) i POLA [24] (Uniwersytet Kalifornijski). TAUM-METEO tłumaczy na użytek publiczny prognozy pogody z angielskiego na francuski. Pozostałe systemy mają charakter eksperymentalny. System POLA, tłumaczący z chińskiego na angielski, zasługuje na uwagę ze względu na fakt, że język chiński nie ma znaków oddzielających słowa i zdania (odstępów i kropek), a system sam rozpoznaje koniec wyrazu i zdania.

Trzecia generacja

Systemy tłumaczenia bezpośredniego i pośredniego korzystają z informacji semantycznej w bardzo ograniczonym stopniu. Tymczasem człowiek przy rozumieniu tekstu korzysta ze znaczenia słów i tekstu, używając swej wiedzy wykraczającej poza czytany opis. Dlatego rozumie również zdanie źle sformułowane gramatycznie (np. zdanie „buty czyścić dziecko" będzie rozumiane jako „dziecko czyści buty", a nie „buty czyszczą dziecko"), nie ma trudności z rozpoznawaniem odwołań zaimkowych („Soldiers fired at women and they fell to the ground" zostanie przetłumaczone jako „żołnierze strzelili do kobiet i one upadły na ziemię", a nie „...i oni upadli na ziemię", gdyż wiemy, że — o ile nie ma odmiennej informacji — osoby trafione kulą padają na ziemię).

Z uwagi na fakt, iż jakość tłumaczeń komputerowych nie jest zadowalająca, rośnie przekonanie, że systemy AT muszą znacznie silniej niż systemy pierwszej i drugiej generacji odwoływać się do semantyki i do ogólnej wiedzy o świecie.

Było to motywem stworzenia przez Wilksa [26] tzw. modelu semantyki preferencyjnej, umożliwiającej w szczególności poprawne przetłumaczenie podanych wyżej przykładów. Na podstawie wiedzy o „normalnej kolei losów" — zawartej w słowniku — system znajduje najbardziej prawdopodobne („preferowane") powiązania logiczne między słowami. System Wilksa tłumaczy z języka angielskiego na francuski przy bardzo ograniczonym słownictwie. Innymi systemami semantycznymi są: LRC Slocuma [21] oraz model zależności koncepcyjnej Carbonella [4], oparty na bazie wiedzy. Propozycja Carbonella zmierza do „zrozumienia" przez system tekstu czytanego w jednym języku, a następnie „opowiedzenia" go w innym języku.

Opisy innych systemów AT, wraz z odsyłaniem do literatury, można znaleźć w przeglądowym artykule [9]. Radzieckie czasopismo „Informatics Abstracts" w każdym numerze poświęca kilka stron aktualnemu rozwojowi systemów AT oraz AS.

METODY TŁUMACZENIA MASZYNOWEGO

W procesie tłumaczenia maszynowego wyróżnia się zasadniczo następujące fazy, które w systemach mogą na siebie częściowo lub całkowicie zachodzić:
● analiza morfologiczna,
● analiza syntaktyczna,
● analiza semantyczna,
● przejście na język docelowy,
● synteza syntaktyczna,
● synteza morfologiczna.

Analiza morfologiczna

Celem analizy morfologicznej jest przygotowanie informacji pomocnej podczas analizy syntaktycznej. Ma ona za zadanie ustalić przynależność pojedynczego słowa do określonej klasy syntaktycznej (część mowy, liczba, przypadek itp.), uzyskując informację wyłącznie z tegoż słowa. W językach o słabej fleksji, np. w angielskim liczba pozyskiwalnej informacji jest szczątkowa, dlatego analiza ma charakter marginalny. Często wszystkie formy wyrazu z informacją morfologiczną umieszcza się bezpośrednio w słowniku, a algorytmy analizy sprowadzają się do obcięcia kilku najczęściej spotykanych typów końcówek (w języku angielskim: -s, -ed, -ing). Inną skrajnością jest język fiński o bardzo rozbudowanej fleksji, gdzie pojedynczy rzeczownik może mieć nawet kilka tysięcy form (a więc i końcówek). W celu przeprowadzenia niezbędnej analizy korzysta się tutaj z metod sztucznej inteligencji. Jäppinen [10] stosuje kaskady wzajemnie wywołujących się reguł, które na podstawie kilku ostatnich liter generują hipotezy gramatyczne, odcinają te litery, resztę wyrazu przekazują innym regułom do dalszej analizy, a następnie do zespołu reguł ustalających hipotezy końcówek form podstawowych (np. bezokolicznika, mianownika liczby pojedynczej). Wygenerowane hipotezy sprawdza się w słowniku form podstawowych.

Analiza morfologiczna jest często niejednoznaczna, tj. dostarcza kilku hipotez dotyczących klasyfikacji syntaktycznej (np. „człowieka" — dopełniacz lub biernik), które mogą być zweryfikowane dopiero podczas analizy syntaktycznej.

Analiza syntaktyczna

Celem analizy syntaktycznej jest ustalenie struktury gramatycznej zdań, tzn. ustalenie podmiotu, orzeczenia, dopełnień itp. oraz ich powiązań. Wejściem do analizy jest tekst źródłowy wraz z informacją morfologiczną. Wynikiem jest na ogół struktura drzewiasta, odzwierciedlająca zależności syntaktyczne.

Oprócz podejścia analogicznego do notacji BNF (znanej z analizy syntaktycznej języków programowania) oraz oprócz dopasowywania (w pierwszych systemach) tekstu do wzorców zdań, w analizie syntaktycznej stosuje się formalizm ATN oraz Q-systemy. Uogólniona sieć przejść (ang. augmented transition network), opisana m.in. w [1], jest rodzajem niedeterministycznego automatu akceptującego kolejno słowa zdania, z możliwością nawrotów w razie niepowodzenia, generującego informację o rozpoznanej strukturze (drzewie). Q-systemy [6], zaliczane do systemów gramatyk transformacyjnych, operują ciągami słów i strukturami drzewiastymi, dokonując ich transformacji, jeżeli pasują do wzorca struktury warunku transformacji.

Analiza semantyczna

Analiza syntaktyczna często nie jest w stanie jednoznacznie określić formy gramatycznej konkretnych słów, wprowadzając dodatkowo wieloznaczność rozbioru gramatycznego zdania. Znalezienie jednoznacznej interpretacji syntaktycznej jest zadaniem analizy semantycznej. Poza tym powinna ona ustalić znaczenie homonimów, np. niemieckie „grosses Tier" znaczy „duże zwierzę", ale jeśli odnosi się do człowieka, to oznacza „grubą rybę". Należy ustalić także, do jakich słów odnoszą się poszczególne zaimki, gdyż tłumaczy się je w zależności od rodzaju słowa, do którego się odnoszą. Przykładowo, niemieckie słowo „Schrank" jest rodzaju męskiego, a jego polski odpowiednik „szafa" — żeńskiego. Tak więc zdania „Eine Katze liegt auf dem Schrank. Sie schläft" nie mogą być słownikowo tłumaczone jako „Kot leży na szafie. Ona śpi". Większość systemów traktuje analizę semantyczną po macoszemu i ogranicza się do podziału słów na klasy semantyczne (rzeczowniki ożywione, nieożywione, ludzie, pojęcia matematyczne) oraz do sprawdzenia, czy zachowane są pewne relacje (np. czyścić mogą tylko ludzie, przedmioty płaskie nie mają objętości itp.). Większość rozwiązań ma charakter doraźny. Systematyczne podejścia proponowali Wilks, Carbonell oraz Slocum.

Przykładowo, metoda Wilksa polega na tym, że każde słowo w słowniku jest opisane typowym jego użyciem (np. pić — POWODOWAĆ poruszanie się CZEGO? płynu PRZEZ CO? czeluść id.). Ponadto do zdań dopasowuje się pewne wzorce, uzupełnia się powiązaniami między wyrazowymi, wynikającymi z typowych „zachowań" opisanych w słowniku. Preferuje się taką interpretację, która najsilniej wiąże elementy zdania i tekstu. W ten sposób odnajduje się zarówno poprawne znaczenie homonimów, jak i odniesienia zaimkowe (aczkolwiek skonstruowano wiele przykładów, dla których metoda Wilksa zawodzi).

Przejście na język docelowy

Oprócz oczywistego sięgnięcia po słownik, przejście na język docelowy oznacza: podstawienie właściwych zaimków oraz właściwego rodzaju przymiotników, zastąpienie fraz (np. „duże zwierzę" — „gruba ryba"), poprawienie struktury syntaktycznej (np. w języku wietnamskim nie ma strony biernej; w polskim stosuje się podwójną negację: np. „I have no time" nie można tłumaczyć jako „Mam żaden czas"), budowę fraz zastępujących nieistniejące słowa (np. w języku niemieckim nie ma przymiotnika „sodowy") itp. Część problemów usuwa odpowiednia analiza semantyczna. Niezbędne przekształcenia strukturalne osiąga się m.in. przez zastosowanie Q-systemów z odpowiednimi zestawami reguł.

Synteza syntaktyczna

Synteza syntaktyczna ma na celu wygenerowanie z docelowej struktury syntaktycznej (na ogół drzewa) tekstu o właściwym dla danego języka stylu w ten sposób, by tekst był zrozumiały i odzwierciedlał sens tekstu źródłowego. Wbrew pozorom jest to zadanie trudniejsze niż analiza, system bowiem odgrywa tu rolę narratora, a nie biernego słuchacza. Na znaczenie fazy zwrócono uwagę stosunkowo niedawno i rozpoczęto prace nad zmodyfikowaniem formalizmu ATN. Krytycznie oceniono podejście systemu CETA, w którym język pośredni „pivot" zatracał pierwotną strukturę zdania — w wielu językach przestawienie szyku wyrazów służy uwypukleniu pewnych myśli.

System Wilksa syntetyzował zdania na podstawie wzorców związanych z poszczególnymi znaczeniami słów w zdaniu. W systemie METEO użyto Q-systemów.

Synteza morfologiczna

Zadaniem syntezy morfologicznej jest nadanie słowom właściwych końcówek odpowiednio do formy deklinacji (koniugacji).

PROBLEMY TŁUMACZENIA MASZYNOWEGO

Jakość tłumaczenia ocenia się według dwóch niezależnych od siebie kryteriów: rzetelności i czytelności, przy czym pierwsze z nich jest decydujące. Kryteria te stosuje się także wobec systemów automatycznego tłumaczenia.

Kłopoty z rzetelnością wynikają z wieloznaczności budowy morfologicznej wyrazów oraz wieloznaczności syntaktycznej zdań. Wieloznaczność morfologiczną można usunąć (choć nie zawsze) przez szczegółową analizę syntaktyczną. Wyeliminowanie wieloznaczności rozbioru syntaktycznego zdania wymaga odwołania się do jego semantyki. Metody analizy syntaktycznej są obecnie dobrze opracowane, lecz analiza semantyczna jest jeszcze w powijakach. Dlatego w wielu systemach zrezygnowano ze szczegółowej analizy semantycznej (która na razie wydaje się możliwa tylko dla bardzo wąskiego słownictwa), ponosząc ryzyko „nieszczęśliwych" tłumaczeń niektórych zdań. Innym wyjściem z nierozstrzygalnych sytuacji jest praca interakcyjna — odwołanie się do człowieka.

Słaba czytelność tekstów AT wynika stąd, że w przeszłości koncentrowano uwagę wyłącznie na analizie tekstu. Dopiero ostatnio, po rozwiązaniu zasadniczych problemów analizy syntaktycznej, zwraca się uwagę na generowanie dobrych tekstów. Jak się bowiem okazuje, do generowania tekstu nie wystarcza proste odwrócenie procesu analizy. Komputer musi dodatkowo przejąć funkcję narratora.

Wspólnym problemem dla systemów słownikowych (AS) i tłumaczących (IT i AT) jest budowa odpowiedniego mechanizmu reakcji na nieznane słowa (pomyłka czy nowe pojęcie?). Z zagadnieniem tym jest związana kwestia uczenia się systemu tłumaczenia na podstawie przetłumaczonych już tekstów (styl, użycie wyrazów w zwrotach itp.). Jak wiadomo, taką zdolność uczenia ma człowiek-tłumacz.

Powyższe problemy pojawiają się również w innych systemach przetwarzania języka naturalnego (systemy pytań i odpowiedzi, systemy ekspertowe z komunikacją w języku naturalnym itp.).

Oceniając przyszły rozwój systemów tłumaczących należy mieć na względzie, że i tak nie wszystkie teksty można przetłumaczyć z języka na język (np. dowcipy odwołujące się do wieloznaczności słów) oraz że rzetelność i wydajność systemów komputerowych należy porównywać z rzetelnością i wydajnością człowieka, który również popełnia fatalne błędy w tłumaczeniu.

ZASTOSOWANIA

Mimo wspomnianych mankamentów i niepowodzeń istnieje wiele systemów, które są z powodzeniem stosowane w takich dziedzinach jak: fizyka jądrowa, loty kosmiczne (projekt SOJUZ-APOLLO), matematyka, oficjalne akty prawne, prognoza pogody, instrukcje obsługi samolotów i inne.

Języki, dla których zbudowano systemy tłumaczące należą do różnych grup językowych, oprócz rosyjskiego, angielskiego, francuskiego, hiszpańskiego i niemieckiego, opracowano systemy tłumaczące na japoński, chiński, turecki, wietnamski, arabski, języki mniejszości narodowych

ZSRR i inne (lub z tych języków). Organizacjami inicjującymi budowę i korzystającymi z AT są wspólnoty międzynarodowe, takie jak: PAHO i EWG, rządy krajów wielojęzycznych, np. ZSRR i Kanady, a także międzynarodowe koncerny.

Systemy tłumaczące na ogół korzystają z komputerów o dużych mocach obliczeniowych i dużych pojemnościach pamięci. Ostatnio podjęto jednak prace nad systemami tłumaczącymi przy użyciu minikomputerów (m.in. w ZSRR).

Zastosowanie systemów AT jest celowe, wówczas gdy:
- dotyczy wąskiej dziedziny, specjalistycznego słownictwa i uproszczonych form gramatycznych,
- istnieje szerokie grono zainteresowanych, nie znających języka oryginału,
- jest niewielu tłumaczy specjalistów w porównaniu z liczbą tekstów do tłumaczenia.

Również współpraca w ramach RWPG sprzyjałaby zastosowaniu systemów automatycznego tłumaczenia, zwłaszcza że odczuwa się brak tłumaczy technicznych.

PERSPEKTYWY AT W POLSCE

Autorowi nie są znane prace zmierzające do budowy konkretnych systemów tłumaczących dla języka polskiego. Nie należy się tutaj spodziewać wypracowania nowych koncepcji tłumaczenia, niezbędne jest natomiast przystosowanie znanych i używanych technik oraz opracowanie metod analizy i syntezy morfologicznej, syntaktycznej i semantycznej. Prace nad tymi zagadnieniami są prowadzone w kraju. Można wymienić prace dotyczące analizy morfologicznej języka polskiego przeprowadzanej przy użyciu specjalnie opracowanego słownika końcówek [11], analizy semantycznej [7], analizy semantycznej fraz przyimkowych [8], implementacji formalizmu ATN z zastosowaniem do analizy syntaktycznej języka polskiego [16].

Podstawowymi trudnościami są:
- luźna struktura zdania w języku polskim, przez co nietypowa będzie analiza syntaktyczna silnie oparta na analizie morfologicznej całego zdania, a nie jego wycinków oraz synteza syntaktyczna, gdyż o kolejności wyrazów we frazach decyduje zrozumiałość tekstu i stopień związku logicznego (np. dzień dobry i dobry dzień),
- duża nieregularność deklinacji i koniugacji, co ma wpływ na rozbudowę analizy morfologicznej i wielkość słownika.

Należy się spodziewać, że coraz bardziej odczuwalny niedobór tłumaczy technicznych — nawet dla tak podstawowych języków jak angielski, rosyjski i niemiecki — spowoduje rozwój narzędzi wspomagających pracę tłumacza. Autorowi znana jest niska jakość tłumaczeń jaką musi się zadowolić wiele instytucji. Z tego też względu budowa praktycznych systemów AT dla języka polskiego mogłaby odegrać pożyteczną rolę.

LITERATURA

[1] Bates M.: The Theory and Practice of ATN Grammers, Pp. 191—260, Natural Language Communication with Computers, L. Bolc (ed.), Lecture Notes in Computer Science, Vol. 63, Springer Verlag, Berlin, 1978
[2] Boitet C.: Mechanical translation and the problem of understanding natural languages. GETA, Grenoble, 1977
[3] Byrne C. E., Scott B. E., Rinth T. H.: Demonstration of LOGOS I system for English-Vietnamese machine translation. Report RADC-TR-70-170 Graffis AFB, Rome Air Development Center, New York, 1970
[4] Carbonell J., Cullingford R., Gershman A.: Knowledge-based machine translation. IEEE Trans. on PAMI, Vol. 3, No. 4, pp. 376—391, 1981
[5] Chandioux J.: METEO — an experimental system for the translation of public weather forecasts. FBIS Seminar on MT, AJCL, pp. 27—36, 1976
[6] Colmerauer A.: Les systemes — Q ou un formalisme pour analyser et synthetiser des phrases sur ordinateur. TAUM 71 ref. 89, pp. 1—45
[7] Dąbrowski M., Dryzek H., Laus-Mączyńska K., Szymanowska Z.: System generowania wyrażeń. Informacja Naukowa — wybrane prace badawcze IINTE za lata 1978—1979. Prace IINTE 34, Warszawa, 1980
[8] Dryzek H.: Analiza semantyczna fraz przyimkowych do celów automatycznego przetwarzania tekstów. Prace IINTE 32, Warszawa, 1980
[9] Hitchins W. J.: Machine translation and machine aided translation. Journal of Documentation, Vol. 14, No. 2, pp. 119—159, 1978
[10] Jäppinen H., Lethola A., Nelimarkka E., Ylilammi M.: Knowledge engineering approach to morphological analysis, Proc. First Conf. of the European Chapter of the ACL, pp. 49—51, Pisa 1983
[11] Jendrzej U.: Automatyczne wyszukiwanie z tekstu wyrazów o ustalonych kategoriach gramatycznych. Praca magisterska, Uniwersytet Śląski, Wydział Techniki, Sosnowiec, 1984
[12] Kay M.: Automatic translation of natural languages. Proc. Am. Acad. Arts Sciences 102, pp. 217—229, 1973
[13] Kay M.: The MIND system. Pp. 155—188, R. Rustin (ed.), Natural language processing. Courant Computer Science Symposium No. 8, 20—21 December 1971, Algorithmic Press, New York, 1973
[14] King M.: EUROTRA — A European system for machine translation. ISSCO, University of Geneva, 1980
[15] Kittredge B.: The Montreal MT prototype. Cahiers Linquistique d'Ottava, No. 2, pp. 69—81, 1972
[16] Kochut K.: Towards the elastic ATN-implementation. The Design of Interpreters, Computers and Editors for Augmented Transition Networks. L. Bolc (ed.), Springer-Verlag, Berlin, 1983
[17] Krollmann F., Schuck H. J., Winkler U.: Herstellung textbezogener Fachwortlisten mit einem Digitalrechner — ein Verfahren der automatischen Übersetzungshilfe. Beiträge zur Sprachkunde und Informationsverarbeitung. Nr. 5, pp. 7—30, 1965
[18] Krollmann F.: Linguistic Data Banks and the technical translator. META, Vol. 16, pp. 117—124, 1971
[19] Languaged and machines in translation and linguistics. A report by the Automatic Language Processing Advisory Committee, Division of Behavioral Sciences, National Academy of Sciences, National Research Council, Washington (DC), 1966
[20] Pan American Health Organization Machine Translation. Summary of Current Status, Washington (DC), 1981
[21] Slocum J., Bunnet W. C.: The LRC machine translation. Working Paper LRC-81-1, University of Texas, Austin, 1982
[22] Toma P.: Computer translation in its own right. Pp. 155—164, Kommunikationsforschung und Phonetik, Buske, Hamburg, 1974
[23] Vangucis B.: La tranduction automatique á Grenoble. Dumond, Paris, 1975
[24] Wang W., Chan S., Tsou B.: Chinese linguistics and the computer. Linguistics, No. 118, pp. 89—117, 1973
[25] Weaver W.: Translation. Machine translation of languages, Locke W. N., Booth A. D. (Eds.), Wiley, New York, 1966
[26] Wilks Y.: An intelligent analyser and understander of English. Comm. of the ACM, Vol. 18, No. 5, pp. 264—274, 1975.

Konferencje

Polioptymalizacja w projektowaniu

W dniach 10—12 czerwca 1988 r. w Mielnie koło Koszalina odbędzie się kolejna **VII Krajowa Konferencja pn. „Polioptymalizacja w projektowaniu"**. Zostanie ona poprzedzona trzy-czterodniową Szkołą, w dniach 6—9 czerwca 1988 r., o podobnej tematyce. Jej organizatorami są: WSI w Koszalinie, Zespół Konstrukcji Maszyn Komitetu Budowy Maszyn PAN oraz ZODOK SIMP w Koszalinie.

Konferencja jest poświęcona problemom teorii, metodom i zastosowaniom optymalizacji i polioptymalizacji w projektowaniu technicznym, a także w procesie podejmowania decyzji w sterowaniu procesami technologicznymi, w zarządzaniu i wszędzie tam, gdzie podejmowane są decyzje na podstawie wielu konfliktowych kryteriów, np. w zadaniach ekonomicznych, transportowych, organizacyjnych itp. Omawiane będą również systemy ekspertowe.

Zadaniem Szkoły będzie nauczanie konkretnych metod oraz formułowania zadań optymalizacji i polioptymalizacji w procesie projektowania oraz zapoznanie z problematyką systemów ekspertowych. Będzie ona przeznaczona dla konstruktorów i projektantów obiektów i procesów technologicznych (np. maszyn, budowli, sieci, procesów przetwórczych, transportowych) oraz dla osób podejmujących decyzje gospodarcze związane z projektowaniem technicznym. W trakcie jej trwania uczestnicy będą rozwiązywać własne problemy z pomocą i radą wykładowców, a do dyspozycji będą komputery klasy IBM PC.

Koszty uczestnictwa przewidywane są na ok. 25 tys. zł (konferencja ok. 10 tys. zł; Szkoła ok. 15 tys. zł). Organizatorzy przewidują możliwość pobytu osób towarzyszących (koszt ok. 15 tys. zł za 6 dni).

Zapytania należy kierować pod adresem:
Ośrodek Doskonalenia Kadr SIMP
ul. Jana z Kolna 38, 75-204 Koszalin, tel. 272-36 w. 7 lub 278-81 w. 239 — doc. dr inż. W. Tarnowski i w. 331 — mgr inż. T. Kiczkowiak.

TADEUSZ KUROCZYCKI
Instytut Maszyn Matematycznych
Warszawa

Redaktor 2000 — mikrokomputerowy procesor tekstów

Każdy system przetwarzania tekstów (procesor tekstu) powinien wykonywać trzy zasadnicze zadania:

- redagowanie dokumentów (wprowadzanie, usuwanie, modyfikowanie tekstu itp.),
- formatowanie dokumentów (ustawianie marginesów, tabulacja, dzielenie tekstu na strony, numerowanie stron, itp.),
- drukowanie dokumentów w formie wybranej przez użytkownika (druk wytłuszczony, druk pochyły, różne wielkości czcionek i odstępy między wierszami).

Użytkownik pragnąc skorzystać z komputerowego systemu przetwarzania tekstów ma do swej dyspozycji wiele najrozmaitszych programów spełniających jego życzenia w węższym lub szerszym zakresie. Chyba każda z większych firm zajmujących się produkcją oprogramowania dla mikrokomputerów opracowała przynajmniej jeden procesor tekstu. Wordstar, Word, Volkswriter, Multimate, PFS Write stanowią tylko przykłady.

Jednakże użytkownik krajowy zanim zaakceptuje dany procesor tekstu, stawia przed nim dodatkowe wymagania. Do najważniejszych z nich należą:

- możliwość redagowania i wyświetlania tekstów w języku polskim (alfabet łaciński rozszerzony o litery: ą, ć, ę, ł, ń, ó, ś, ż, ź),
- możliwość drukowania tekstów w języku polskim na sprzęcie produkcji krajowej.

Procesorem tekstów spełniającym powyższe warunki jest Redaktor 2000, działający na komputerze Mazovia 1016 oraz na mikrokomputerach kompatybilnych z IBM PC.

PODSTAWOWE WŁAŚCIWOŚCI PAKIETU

Redaktor 2000 jest ekranowo ukierunkowanym pakietem oprogramowania spełniającym wspomniane trzy podstawowe funkcje procesora tekstów, stanowi produkt programowy w polskiej wersji językowej. Pozwala na swobodne użycie przez osobę nie będąca programistą oraz nie władającą językiem obcym, bowiem wszystkie teksty wyświetlane przez program na ekranie: nazwy i spisy poleceń, komunikaty o błędach, wskazówki dla użytkownika są zapisane w języku polskim. Umożliwia pełne redagowanie tekstów, w których występują litery alfabetu polskiego.

Do nośnika zawierającego postać programową pakietu jest dołączana dokumentacja użytkowa w języku polskim, zawierająca — oprócz opisu funkcjonalnego samego pakietu — omówienie różnych wariantów jego instalowania i eksploatacji, w zależności od konfiguracji dostępnej użytkownikowi.

Mgr TADEUSZ KUROCZYCKI ukończył w 1971 r. studia na Wydziale Matematyki i Mechaniki Uniwersytetu Warszawskiego. Jest adiunktem w Instytucie Maszyn Matematycznych w Warszawie. Uczestniczył w pracach związanych z systemami operacyjnymi maszyn Jednolitego Systemu, zajmując się problematyką teleprzetwarzania i procesami sieciowymi. Obecnie bierze udział w pracach nad produktami programowymi dla maszyn SM, m.in. w przygotowaniu oprogramowania dla mikrokomputera MAZOVIA 1016.

```
TEST2.RD2      Str. 1  Wie 1 Kol 1          Wstaw  Poz
                    R E D A G O W A N I E
+-------------------------------------------------------+
| ^Bloki      ^Taby i marginesy   ^Parametry drukowania |  ^Na pomoc
| ^Kursor     ^L szukaj           ^Usuń       ^Cofnij   |  ^O-Koniec
| ^Opcje      ^Jeszcze szukaj     ^Y Skróty             |
+-------------------------------------------------------+
          ^O - przytrzymaj Ctrl i naciśnij O.
|---v---1v---v---2v---v---3v---v---4v---v---5v---v---6v---|

Warszawa, &data&

                                   Kol. &adresat&
                                   ul. &ulica& m &mieszkanie&
                                   &kod&

                    Z A P R O S Z E N I E

W dniu 31 lipca 1987r. w Klubie PTTK odbędzie się prelekcja
mgr Krzysztofa Wilińskiego pt.

   "Moje wrażenia z pobytu w Nepalu".

Prelekcja będzie ilustrowana przeźroczami. Wstęp wolny.


Maria Kowalik,Senatorska 45,24,00-116 Warszawa
Andrzej Wąsowski,Prosta 21,3,00-245 Warszawa
Marek Lipiński,Freta 66,26,00-456 Warszawa
```

Rys. 1. Postać sparametryzowanego dokumentu oraz danych wykorzystywanych podczas drukowania kolejnych wersji tego dokumentu

Redaktor 2000 umożliwia tworzenie, przechowywanie, modyfikowanie i drukowanie plików tekstowych. Pliki mogą być kopiowane, usuwane i łączone. Pakiet zapewnia dodawanie i usuwanie tekstu, wyszukiwanie (w dowolnych kierunkach) i ewentualne wymienianie wybranych sekwencji. Podczas redagowania kursor może swobodnie przesuwać się po ekranie, istnieje też możliwość operowania wybranymi fragmentami obrazu (np. usuwanie).

Z formatowaniem są związane m.in. następujące możliwości: określanie długości strony, ustawianie marginesów (górnego, dolnego, bocznych) i odstępów między wierszami, automatyczne przenoszenie wyrazów, dosuwanie tekstu, ustawianie tabulacji. Na rysunku 1 przedstawiono postać sparametryzowanego dokumentu oraz danych wykorzystywanych podczas drukowania.

W przypadku drukowania tekstu użytkownik może zażądać podkreślenia lub wytłuszczenia poszczególnych fragmentów, automatycznego numerowania stron itp. Istnieje możliwość uzyskiwania druku różnego rodzaju (większe litery, mniejsze litery, zróżnicowane odstępy pomiędzy wierszami, druk pochyły itp.). Redaktor 2000 obsługuje większość typów drukarek zagranicznych oraz polską drukarkę D100 PC. Na rys. 2 przedstawiono wydruk dokumentu połączony z zastosowaniem różnych czcionek oraz możliwości wymienionych w następnym punkcie.

SPECJALNE WŁAŚCIWOŚCI FUNKCJONALNE

Oprócz standardowych funkcji spełnianych przez większość procesorów tekstów, Redaktor 2000 ma wiele specjalnych właściwości funkcjonalnych.

- **Równoległe redagowanie do trzech fragmentów tekstu za pomocą podziału ekranu na okna**

Okno jest wygodnym narzędziem do redagowania kilku dokumentów naraz lub do równoczesnego oglądania kilku

miejsc w jednym dokumencie. Otwarcia okna dokonuje się za pomocą polecenia ^OO, a zamknięcia za pomocą polecenia ^QP. Między oknami można wykonywać operacje kopiowania (^BS) i przenoszenia bloków (^BT).

Warszawa, 15 lipca 1987r.

Kol. Maria Kowalik
ul. Senatorska 45 m 24
00-116 Warszawa

ZAPROSZENIE

W dniu 31 lipca 1987r. w Klubie PTTK odbędzie się prelekcja mgr Krzysztofa Wilińskiego pt.

"Moje wrażenia z pobytu w Nepalu".

Prelekcja będzie ilustrowana przeźroczami. Wstęp wolny.

Warszawa, 15 lipca 1987r.

Kol. Andrzej Hąsowski
ul. Prosta 21 m 3
00-245 Warszawa

ZAPROSZENIE

W dniu 31 lipca 1987r. w Klubie PTTK odbędzie się prelekcja mgr Krzysztofa Wilińskiego pt.

"Moje wrażenia z pobytu w Nepalu".

Prelekcja będzie ilustrowana przeźroczami. Wstęp wolny.

Warszawa, 15 lipca 1987r.

Kol. Marek Lipiński
ul. Freta 66 m 26
00-456 Warszawa

ZAPROSZENIE

W dniu 31 lipca 1987r. w Klubie PTTK odbędzie się prelekcja mgr Krzysztofa Wilińskiego pt.

"Moje wrażenia z pobytu w Nepalu".

Prelekcja będzie ilustrowana przeźroczami. Wstęp wolny.

Rys. 2. Wynik drukowania dokumentu przedstawionego na rys. 1

● **Łączenie fragmentów różnych dokumentów.**
Redaktor 2000 umożliwia wstawianie całości lub fragmentów wybranych dokumentów do innych tekstów. Temu celowi służą odpowiednio polecenia ^BA oraz ^BD.

● **Wykonywanie operacji arytmetycznych w trakcie redagowania**
Dopuszczalne jest wykonywanie podstawowych operacji arytmetycznych, jak: dodawanie, odejmowanie, mnożenie, dzielenie i potęgowanie. Operacje te mogą być wykonywane w wierszach lub w kolumnach. Aby wykonać działanie arytmetyczne, należy skorzystać z polecenia ^BO.

● **Pisanie listów i ich adresowanie**
Redaktor 2000 dostarcza aparatu programowego służącego do tworzenia sparametryzowanych dokumentów, przystosowanych do drukowania w różnych wariantach. Dokument taki zawiera zmienne (ciąg alfanumeryczny ograniczony symbolami &), którym wartości są przyporządkowywane bądź na bieżąco, bądź też z uprzednio przygotowanego (też za pomocą Redaktora 2000) pliku danych. Najczęściej parametryzowanymi dokumentami są listy. Fragment sparametryzowanego dokumentu przedstawiono na rys. 1.

● **Odzyskiwanie uprzednio usuniętego tekstu**
Polecenie cofania (^C) powoduje przywrócenie tekstu, który został ostatnio usunięty przy użyciu polecenia blokowego usuwania (^BU) lub usuwania (^U). Usunięty tekst jest przechowywany do chwili powrotu do wstępnego menu (rys. 3).

● **Stosowanie skrótów**
Możliwe jest definiowanie (polecenie ^YD), a następnie wykorzystywanie skrótowych form tekstu, które będą automatycznie zastępowane w redagowanym dokumencie przez odpowiadający im dłuższy tekst. Dopuszcza się również korzystanie z innych plików, zawierających definicje skrótowych postaci tekstu (polecenie ^YZ).

WSTĘPNE MENU

Redaguj	Drukuj	Na pomoc
Usuń	Kopiuj	Q-koniec
Wybierz stację	Y Skróty	
Przenieś	Format	
Tryb pisania na maszynie		

Wybierz podświetloną literę

Rys. 3. Wstępne menu Redaktora 2000

● **Wykorzystywanie komputera jako maszyny do pisania**
Wydanie polecenia T powoduje przejście do trybu pisania na maszynie. Możliwe są dwa warianty — bez buforowania lub z buforowaniem. W pierwszym przypadku, aż do odwołania, sekwencje tekstu wprowadzane z klawiatury są od razu wyprowadzane na drukarkę. W drugim przypadku, tekst jest buforowany i po zwolnieniu wyprowadzany na drukarkę. Buforowanie umożliwia podręczną korektę tekstu.

POLECENIA PAKIETU

Użytkownik pakietu ma do swej dyspozycji następujące polecenia realizujące podstawowe operacje związane z przetwarzaniem tekstu:
^B — operowanie blokami tekstowymi,
^C — anulowanie ostatnich modyfikacji tekstu,
D — drukowanie,
F — określanie formatów,
K — kopiowanie,
^L — poszukiwanie określonej sekwencji tekstowej,
^O — wybieranie sposobu redagowania,
^P — określanie trybu drukowania,
R — redagowanie tekstu,
^T — ustawianie tabulacji i określania marginesów,
^U — usuwanie tekstu,
^Y — posługiwanie się skrótami.
Większość powyższych poleceń ma pewną liczbę podpoleceń odpowiadających poszczególnym wariantom działania.

Wydanie podpolecenia polega na wprowadzeniu sekwencji: ^ <**identyfikator podpolecenia**>
Przykładowo, do podpoleceń polecenia ^O (opcje) należą:
^A — określanie stopki (tekstu powtarzającego się na dole drukowanych stron),
^C — umieszczanie tekstu na środku wiersza (centrowanie),
^F — zmienianie sposobu wyświetlania tekstu (ze znakami tabulacji, znacznikami poleceń, przypisami i komentarzami lub bez nich),
^G — określanie paginy (tekstu powtarzającego się u góry drukowanych stron),
^J — włączanie i wyłączanie automatycznego dosuwania tekstu,
^K — umieszczanie komentarza w tekście,
^O — otwieranie nowego obszaru roboczego na ekranie (okna),
^P — definiowanie przypisów oraz odsyłaczy do przypisów,
^R — powtarzanie poleceń lub tekstu ,
^S — przechodzenie do nowej strony,
^T — parametryzowanie tekstu tworzenie dokumentu przystosowanego do drukowania w różnych wersjach),
^U — określanie nowego numeru aktualnej strony,
^W — tworzenie grupy wierszy,
^Z — przełączanie trybu przebijania na tryb wstawiania i odwrotnie,
^— — określanie dozwolonego miejsca przenoszenia w wyrazie.

Podczas działania pakietu użytkownik wybiera polecenia posługując się wyświetlanym na ekranie menu.

Bezpośrednio po rozpoczęciu pracy wyświetlane jest wstępne menu (rys. 3). Z tego menu użytkownik wybiera

jedną z podstawowych funkcji pakietu, przez wprowadzenie odpowiadającego jej rozjaśnionego identyfikatora. Po wyborze podstawowej funkcji pakietu wyświetlane są menu następnego poziomu, jak **WYBÓR NAZWY** lub **DECYZJE**.

Podstawową funkcją pakietu Redaktor 2000 jest redagowanie tekstów. Menu **REDAGOWANIE** zawiera zestaw operacji potrzebnych do wykonywania prac redakcyjnych. Podczas realizowania poszczególnych operacji redakcyjnych użytkownik korzysta z menu operacyjnych, takich jak np. **BLOKI** czy **OPCJE**.

W trakcie działania użytkownik może, w dowolnym momencie, skorzystać z informacji pomocniczych związanych z aktualnie wykonywanym poleceniem. Aby te informacje uzyskać należy wydać polecenie ^N. Na rysunku 4 przedstawiono postać informacji pomocniczych dotyczących określania parametrów drukowania [1]).

NA POMOC -PARAMETRY DRUKOWANIA cz.1

Naciśnij ^P i pierwszą literę komendy przed oraz po tekście, którego komenda ma dotyczyć. Litera pojawi się w nawiasach, np: [B].

Komenda	Odpowiedź
Pogrubienie	Drukuje tłustym drukiem.
Wybicie	Zależy od typu drukarki. Może np. podwójnie pogrubić tekst.
Zamazanie	Przekreśla znaki kreską poziomą przy drukowaniu.
L podkreślenie	Podkreśla tekst łącznie z przerwami między słowami, o ile to nie zmienione w formacie dokumentu.
+ superskrypt	Drukuje powyżej linii wiersza.
- subskrypt	Drukuje poniżej linii wiersza.

Wprowadź SPACJE dla dalszej pomocy, ESC dla powrotu.

Rys. 4. Fragment informacji pomocniczych dotyczących parametrów drukowania

PORÓWNANIE Z INNYMI PROCESORAMI TEKSTÓW

Krajowi użytkownicy mogą już korzystać z kilku procesorów tekstu opracowanych w polskiej wersji językowej (mowa, oczywiście, o programach działających na mikrokomputerach kompatybilnych z IBM PC). Oferują je m.in. firmy CSK i Computex.

Jak na razie, Redaktor 2000 przewyższa je pod względem możliwości funkcjonalnych. Chodzi tu zwłaszcza o omówione powyżej tzw. specjalne właściwości procesora. W takie udogodnienia, jak np. praca w kilku oknach, odwoływanie wprowadzonych modyfikacji, przeszukiwanie i wielokrotne zamienianie jednej sekwencji tekstowej na inną, są wyposażone tylko nieliczne procesory tekstowe firm zachodnich, jak Word, czy Visiword 1.2.

Pod względem wymagań sprzętowych Redaktor 2000 nie odbiega od poziomu innych produktów tego rodzaju. Wymagania te — minimum 256 KB RAM i dwie stacje dyskietek 5,25 cala lub jedna stacja dyskietek oraz dysk stały — są spełniane przez typową konfigurację komputerową kompatybilną z IBM PC.

Redaktor 2000 stanowi cały podsystem przetwarzania tekstów. Konsekwencją znacznego rozszerzenia funkcjonalnego są nieco gorsze parametry czasowe.

Jeśli chodzi o dalsze funkcjonalne rozszerzanie Redaktora 2000, to na ukończeniu znajduje się wersja umożliwiająca wykonywanie wielu dodatkowych prac redakcyjnych. Chodzi tu m.in. o:
— tworzenie spisów treści, indeksów i nagłówków dla poszczególnych pozycji wydawniczych,
— budowanie kartotek, z których można — w prosty sposób — pobierać informacje przydatne podczas redagowania pism kierowanych do instytucji, czy osób prywatnych. Powyższe, specjalistyczne możliwości posiadają tylko bardzo nieliczne procesory tekstów.

W dalszej przyszłości przewiduje się wbudowanie do Redaktora 2000, bardziej uniwersalnego niż obecny samouczka (ang. help). Byłby on w stanie udzielać bardziej szczegółowych wyjaśnień i wskazówek — w zależności od konkretnych potrzeb użytkownika.

[1]) Treść tych informacji i sposób korzystania z nich jest charakterystyczny dla podręcznika zwanego samouczkiem. Proponujemy więc przyjęcie tego, dość naturalnego, odpowiednika polskiego na oznaczenie angielskiego terminu help (przyp. red.).

Konferencje

Systemy Zarządzania Bazami Danych

Kolejne, 11. Międzynarodowe Seminarium nt. Systemów Zarządzania Bazami Danych odbędzie się na Węgrzech w pięknym zamku w miejscowości Seregélyes (w połowie drogi między Budapesztem a jeziorem Balaton) w okresie od 3 do 7 października 1988 r. Celem 11. Seminarium, podobnie jak jego poprzedników, jest rozpoznanie i zachęcenie do badań, rozwoju i zastosowań w dziedzinie systemów zarządzania bazami danych i w dziedzinach pokrewnych. Będzie ono stanowić okazję dla praktyków i naukowców z różnych krajów do wymiany poglądów na temat najnowszych osiągnięć w dziedzinie zarządzania bazami danych, obejmujących nowe metody, działalność praktyczną, narzędzia programowe i aplikacje.

Głównymi (lecz nie jedynymi) tematami konferencji będą:
- Architektura systemów baz danych,
- Teoria i modele baz danych,
- Metodologie i narzędzia do projektowania baz danych, rozproszone bazy danych,
- Zaawansowane oprogramowanie użytkowe.

Szczególnie mile będą widziane prace dotyczące nowych dziedzin, takich jak:
- Systemy z bazą wiedzy,
- Logiczne bazy danych,
- Bazy danych dla komputerowego wspomagania projektowania i produkcji,
- Wielonośnikowe bazy danych.

Na program naukowy Seminarium złożą się referaty znanych autorów, zamówione przez organizatorów, najlepsze spośród nadesłanych artykułów oraz dyskusje panelowe. Dla najlepszego referatu Seminarium, wybranego z pomocą uczestników, jest przewidziana specjalna nagroda.

Zgłoszenia zamiaru uczestnictwa i/lub wygłoszenia referatu oraz sugestie dotyczące programu należy nadsyłać do dnia 31 stycznia 1988 r. pod adresem Sekretarza Komitetu Programowego Ms. Olga Szilvássy, Központi Statisztikai Hivatal, 11th
ISDBMS, Budapest, Pf.51., H-1525, Hungary (tel. 152-850 w. 167 lub 35-716, telex 22 4308)
oraz pod adresem właściwego krajowego członka Komitetu Programowego (dla autorów z Polski jest nim Władysław Bogucki, CPiZI ZETO-ZOWAR, Al. Niepodległości 190, 00-608 Warszawa).

Autorzy proszeni są o nadsyłanie pełnego tekstu oryginalnych artykułów (maszynopis w języku angielskim, z podwójnymi odstępami między wierszami, maksymalnie 20 stron), odzwierciedlających najnowsze wyniki ich pracy. Jedną kopię artykułu powinien otrzymać do dnia 15 marca 1988 r. Sekretarz Komitetu Programowego, a drugą — do dnia 31 maja 1988 r. — właściwy krajowy członek Komitetu Programowego. W terminie do dnia 31 maja 1988 r. autorzy otrzymają zawiadomienie o przyjęciu bądź odrzuceniu referatów. Referaty w ostatecznej formie (maszynopisy z pojedynczymi odstępami między wierszami, maksymalnie 12-stronicowe, nadające się do kopiowania metodą fotograficzną), w których zostały uwzględnione zasugerowane zmiany powinny być nadesłane do dnia 15 lipca 1988 r.

Materiały konferencyjne zostaną opublikowane przez znaną oficynę wydawniczą.

Liczba uczestników Seminarium jest ograniczona. Pierwszeństwo mają kandydaci, którzy będą aktywnie uczestniczyć w realizacji programu. Zasadniczo zakłada się, że uczestnicy będą pokrywać swoje wydatki z własnych źródeł. Młodzi autorzy (do 28 lat), którzy zaprezentują prace na wysokim poziomie, mogą jednak liczyć na finansowe wsparcie ze strony organizatorów. W związku z tym wysyłając swoje prace powinni zaznaczyć, czy z takiej możliwości chcieliby skorzystać.

Opracował:
WIKTOR RZECZKOWSKI

Spis treści rocznika

INFORMATYKA 1987

ARTYKUŁY PROBLEMOWE

	nr	str.
AMBROZIAK JACEK — Systemy ekspertowe		
Część 2	1	19
BIELECKI JAN — Język C Implementacja Lattice/Microsoft	1	17
Definiowanie funkcji ze zmienną liczbą argumentów	8	19
Wykreślanie podstawowych obiektów graficznych	9	18
Programowanie operacji wejścia-wyjścia	10	17
Konstruowanie, interpretowanie i wyznaczanie wartości wyrażeń	11—12	37
BIELECKI JAN — Turbo Pascal	7	10
BIELECKI JAN — Podstawy grafiki w języku Turbo-Pascal		
Część 1. Przegląd środków graficznych	8	5
Część 2. Wykreślanie podstawowych obiektów graficznych	9	14
Część 3. Zarządzanie oknami i grafika żółwia	10	11
BIEŃKOWSKI KAZIMIERZ — Systemy przetwarzania tekstów	11—12	11
CELLARY WOJCIECH — Systemy mikroprocesorowe		
Część 2	1	10
COFTA PIOTR — Czy następuje zmierzch języków asemblerowych?	3	8
DWORAKOWSKI WALDEMAR (oprac.) — Szczególnie istotne adresy, przerwania i porty IBM PC/XT	4	21
GONDZIO MAREK — Sieci Petriego		
Część 1. Definicja i analiza własności sieci	1	5
Część 2. Przekształcenia i rodzaje sieci	2	13
GOOS GERHARD — Nowoczesne języki wysokiego poziomu a produkcja oprogramowania	9	1
GÓRAL STANISŁAW, JANCZEWSKI KRZYSZTOF, JANYSZEK JÓZEF, ZAJĄCZKOWSKI ZBIGNIEW — Użytkowanie Międzyuczelnianej Sieci Komputerowej	3	15
GRABOWICZ D. (oprac.) — Wprowadzenie do programowania w języku asemblera IBM PC		
Część 1	9	20
Część 2	10	19
GRABOWICZ ROMAN, ZALEWSKI JANUSZ — System operacyjny PC-DOS		
Część 4	1	21
Część 5	2	17
GRUDZIŃSKI WŁODZIMIERZ — Zastosowanie Prologu w bazach danych		
Część 1	8	1
Część 2	9	11
GRZYWAK ANDRZEJ, KOŁODZIEJ JERZY — Sieci komputerowe oparte na mikrokomputerach MERA 60	3	11
GURBIEL EWA, KRUPICKA HELENA, PŁOSKI ZDZISŁAW — Składnia języka Logo — próba formalizacji		
Część 1	4	5
Część 2	5	10
JONES CLIFF B. — Specyfikacje a programy		
Część 1. (tłum. i oprac. Wojciech Pachocki)	10	1
Część 2. (tłum. i oprac. Wojciech Pachocki)	11—12	2
KŁOPOTEK MIECZYSŁAW — Maszynowe tłumaczenie tekstów	11—12	20
KOWALSKI ROBERT — Programowanie logiczne i komputery nowej generacji		
Część 1. (tłum i oprac. Edmund Pierzchała)	3	2
Część 2. (tłum. i oprac. Edmund Pierzchała)	4	8
KOWALSKI ROBERT — Logika w systemach ekspertowych		
Część 1. (oprac. i tłum. Marek Machura)	5	1
Część 2. (oprac. i tłum. Marek Machura)	6	5
KUC MARIUSZ (oprac.) — Kompilator Ady dla mikrokomputerów	6	20
KUC M. (oprac.) — Norma ANSI języka C	8	20
KUC M. (oprac.) — Przykład wielozadaniowości w Adzie	10	23
KUROCZYCKI TADEUSZ — Redaktor 2000 — mikrokomputerowy procesor tekstów	11—12	24
ŁUKASZEWICZ WITOLD — Przechowywanie wiedzy i niewiedzy w bazach danych		
Część 1	2	1
Część 2	3	5
MADEY JAN — Problematyka systemów operacyjnych na przykładzie systemu Unix		
Część 1	6	1
Część 2	7	8
MELER KAPCIA MARIA — System relacyjnej bazy danych SELKO dla minikomputera Mera 400	4	23
MOKRZYCKI WOJCIECH — System programowania grafiki komputerów PSG 2.2	3	23
MORZY TADEUSZ — Synchronizacja transakcji w systemach rozproszonych baz danych		
Część 1	5	7
Część 2	6	11
OSSOWSKA STANISŁAWA — Wordstar 3.30 — zasady działania i sposób użytkowania		
Część 1	9	9
Część 2	10	15
Część 3	11—12	30

	nr	str.
OWCZARCZYK JACEK, STOLARSKI MACIEJ, WOŹNIAK EWA — Procesory tablicowe — współbieżne systemy przetwarzania obrazów	2	8
PAWŁOWSKI MAREK — Programator pamięci UVEPROM serii 27XXX	2	20
PAWŁOWSKI MAREK — Mikroprogramowanie		
Część 1	4	11
Część 2	5	12
PŁOSKI ZDZISŁAW (oprac.) — Język awk — przetwarzanie wzorców	11—12	16
PŁOSKI ZDZISŁAW — Chiwriter — procesor tekstu		
Część 1	11—12	27
RATKA CZESŁAW — Adaptacja drukarki wierszowej DW 204-2 do pracy z minikomputerem SM-4	6	14
RAZNOWIECKI ANDRZEJ, SYFERT ANNA — Funkcje systemu BIOS dla PC/XT	4	15
RAZNOWIECKI ANDRZEJ, SYFERT ANNA, ZALEWSKI JANUSZ (oprac.) — Struktura systemu operacyjnego PC-DOS		
Część 1	7	17
Część 2	8	16
Część 3. (autorzy oprac.: Raznowiecki Andrzej, Syfert Anna)	11—12	32
ROKITA PRZEMYSŁAW — Nowa wersja programu szeregującego wysokiego poziomu dla systemu operacyjnego George 3	3	25
RZECZKOWSKI WIKTOR (oprac.) — FILEPRO 16 i FILEPRO 16 PLUS — Oprogramowanie do zarządzania bazą danych pod nadzorem systemu MS-DOS lub Unix	7	22
SHAW MARY M. — Poza programowanie wielkoskalowe — kolejne wyzwania inżynierii oprogramowania		
Część 1. (tłum. i oprac. Janusz Zalewski)	7	1
Część 2. (tłum. i oprac Janusz Zalewski)	8	8
SKOLIMOWSKI JANUSZ — Specjalizowany system mikrokomputerowy SIGMA-3	2	12
STANISZKIS WITOLD — Zarządzanie rozproszonymi danymi — przegląd problematyki		
Część 1. (tłum. i oprac. Wiktor Rzeczkowski	10	5
Część 2. (tłum. i oprac. Wiktor Rzeczkowski)	11—12	7
STRYCHARCZYK JERZY (oprac.) — Janus/Ada — użyteczne narzędzie do nauczania Ady	5	20
SZKARADNIK ZBIGNIEW — Grafika żółwia w języku Forth		
Część 2	6	8
TABAK DANIEL — Intel 80386 i nowe mikroprocesory 32-bitowe		
Część 1. (tłum. i oprac. Janusz Zalewski)	1	1
Część 2. (tłum. i oprac. Janusz Zalewski)	2	4
TADEUSIEWICZ RYSZARD — Głosowe wprowadzanie informacji do komputera		
Część 1	7	5
Część 2	8	12
TRAWIŃSKI GRZEGORZ — Pakiet zintegrowany COMPUTEX	3	19
WĘGLARZ JAN — Ocena działania systemów komputerowych — przedmiot — metody, kierunki rozwoju		
Część 1	4	1
Część 2	5	4
WRYCZA STANISŁAW — Podejścia do tworzenia schematów konceptualnych	1	13
ZALEWSKI JANUSZ — Nasze miejsce na mapie	3	1
ZALEWSKI JANUSZ (oprac.) — Porównanie systemów BIOS dla rodziny komputerów IBM PC	4	18
ZALEWSKI JANUSZ (oprac.) — Concurrent DOS		
Część 1	5	18
Część 2	6	17
ZALEWSKI JANUSZ (oprac.) — Obszar komunikacyjny system BIOS	7	20
ZALEWSKA M. M. — Orkiestra na pięć fajerek czyli niezależna ocena pakietu Computex	3	22
ZIELCZYŃSKI PIOTR — MULTICOMP — system rozwiązywania zadań metodą przeszukiwania drzew		
Część 1	9	5
Część 2	10	8

DYDAKTYKA

Program nauczania informatyki w małych uczelniach amerykańskich — (oprac.) **Paweł Grzegorzewicz**	1	24
Nauczanie informatyki — **Ryszard Tadeusiewicz**	2	22

SAMOTESTY

V.A-C. Bazy danych — oprac. **Adam B. Empacher, Jerzy L. Rossowski**	4	24
V.D. Podejście hierarchiczne w projektowaniu baz danych — oprac. **Adam B. Empacher, Jerzy L. Rossowski**	7	24

Z KRAJU

Programista zawodowy i wartość jego pracy — **Zdzisław Płoski**	1	26
Poprawa efektywności testowania wymienności taśm magnetycznych w systemach Odra 1300 — **Krzysztof Lisiak**	1	28
Systemy mikrokomputerowe w zarządzaniu gospodarką — **Maciej Adamczyk**	3	27
Układ pracy naprzemiennej dla DZM 180--KSRE — **Stanisław Wyszyński**	5	23
Konkurs PTI na najlepsze prace magisterskie z informatyki rozstrzygnięty — **Czesław Daniłowicz, Zbigniew Szpunar**	7	25
II Walny Zjazd Polskiego Towarzystwa Informatycznego — **Jarosław Deminet**	10	25

ZE ŚWIATA

Wielodostępne systemy operacyjne dla IBM PC/XT — oprac. **Piotr Strutyński**		
Część 1	1	30
Część 2	2	25
Targi Lipskie'86 — **Władysław Klepacz**	2	28
IBM — numer 1? — oprac. **Wacław Iszkowski**	3	28

	nr	str.
Janus/Ada — podzbiór Ady dla mikrokomputerów — oprac. **Jerzy Strycharczyk**	4	26
Kostki z arsenku galu — oprac. **Jan Ryżko**	4	28
Stan usług w dziedzinie banków danych w Japonii — **Wiktor Rzeczkowski**	4	30
Tłumaczenie maszynowe — oprac. **M. Machura**	4	30
RISC czyli komputer uproszczony — **Dorota Inkielman**	5	26
Komputery optyczne — oprac. **M. Machura**	5	28
Systemy ekspertowe na mikrokomputery IBM PC — oprac. **M. Machura**	6	25
Kompilatory Ady na komputery IBM PC — oprac. **Mariusz Kuc**	7	26
Kto jest najszybszy — **M. Machura**	7	29
Programowanie w Adzie rozproszonych systemów docelowych — oprac. **M. Kuc**	8	25
Pierwszy mikrokomputer z mikroprocesorem 80386 — oprac. **Jan Ryżko**	8	26
FASTBUS — modularny system magistralowy — **K. Rzymkowski**	9	23
Ada — konwencje kodowania — oprac. **M. Kuc**	9	24
Krzemowy Forth czyli procesor NOVIX NC 4000 — **Zbigniew Szkaradnik**	10	27
Głosowy dostęp do baz danych — oprac. **M. Machura**	11—12	43
CeBIT '87 — **Władysław Klepacz**	11—12	46
Systemy ekspertowe w procesach technologicznych — oprac. **Piotr Strzałkowski**	11—12	44

Kto jest kim w IFIP

Blagovest H. Sendov	4	29
Heinz Zemanek	6	28
Jacques Fourot	7	30
Graham Morris	9	27

Kto jest kim

Jean D. Ichbiah — oprac. **Mariusz Kuc**	11—12	47

TERMINOLOGIA

	nr	str.
Terminologia lokalnych sieci komputerowych — **Janusz Zalewski**		
Część 2	1	III okł.
Część 3	2	29
Część 4	3	31
Polskie nazewnictwo znaków kodu ASCII/ISO Część 2	4	31
Terminologia polskiego Logo — **Janusz Zalewski**		
Część 1	5	30
Część 2	6	29
Część 3	7	III okł.
Metakody znaków alfanumerycznych — **Wacław Iszkowski**	9	28
Słownik IBM PS/2	10	24
Język C — propozycja polskiej terminologii — **Jan Bielecki**	11—12	39

KONFERENCJE

	nr	str.
Convention Informatique '87	1	9
Euromicro '87	1	27
Wielkie Bazy Danych (VLDB)	2	3
CeBIT '87	2	11
VI Krajowa Konferencja „Polioptymalizacja w projektowaniu"	3	21
Systemy Zarządzania Bazami Danych	4	4
COLING '88	5	6
SIGGRAPH '87	5	17
EUROMICRO '87. Mikroarchitektury, rozwój i zastosowanie — **Adam Pawlak**	5	26
SYSTEMS '87	6	4
Techniki komputerowe w zarządzaniu produkcją	6	24
Entity — Relationship Approach '86 — **W. Rzeczkowski**	6	27
Zjazd Użytkowników komputerów Sharp	6	III okł.
INFOGRYF '88	11—12	10
Polioptymalizacja w projektowaniu	11—12	23
Systemy Zarządzania Bazami Danych	11—12	26
INFOPRO '87 — **Maciej Robakiewicz**	11—12	41
INFOSYSTEM '87 — **Jarosław Deminet**	11—12	42

RECENZJE

	nr	str.
Komputerowe projektowanie systemów baz danych — **Piotr J. Jasiński, Zbyszko Królikowski, Jacek Szulczyński**	5	29
Drugie wejście Codda — **Piotr J. Jasiński**	8	24

POGLĄDY

	nr	str.
Logo Sinclaira i jego modyfikacje — przyczynek do dyskusji o jakości pozasprzętowej bazy szkolnej edukacji informatycznej — **Ewa Gurbiel, Helena Krupicka, Zdzisław Płoski**	8	28
Procesory Intela — inny punkt widzenia — **Michał Choroszucha**	10	31

UWAGA UŻYTKOWNICY KOMPUTERÓW SM-4 I MERA 60/660

Już do nabycia system graficzny KOLORGRAF K-640/QK-640 przeznaczony do współpracy z komputerami serii SM i MERA 60/600.

System umożliwia:

- tworzenie i wyświetlanie różnorodnych informacji graficznych,
- realizację podstawowych operacji graficznych, jak kreślenie wektorów, znaków alfanumerycznych, wykresów funkcji itp.

System K-640/QK-640 jest rastrowym systemem graficznym i może mieć wiele zastosowań w grafice komputerowej m.in. w komputerowym wspomaganiu projektowania (CAD), monitorowaniu złożonych procesów, animacji, wzornictwie.

System KOLORGRAF K-640/QK-640 składa się z:

- sterownika graficznego typu KG 640-SM/QKG 640
- monitora graficznego GVT-640 o wysokiej rozdzielczości.

Zamówienia prosimy kierować pod adres:

CBW MERCOMP Sp. z o.o. ul. Poezji 19, 04-994 WARSZAWA

mercomp Sp. z o.o.

Bliższe informacje można uzyskać pod telefonem 12-90-11 w. 1074

EO/831/87

MIĘDZYWOJEWÓDZKA SPÓŁDZIELNIA PRACY „SIÓDEMKA"

ŁÓDŹ, AL. KOŚCIUSZKI 93,

ZAKŁAD INFORMATYKI I SYSTEMÓW KOMPUTEROWYCH

poleca po **najniższych cenach** w kraju:

— mikrokomputery 8-, 16-, 32-bitowe najwyższej jakości renomowanych firm z całego świata
— urządzenia peryferyjne:
- drukarki — również 24-igłowe i laserowe
- streamery
- napędy dyskowe 3", 5.25"
- monitory monochromatyczne, kolorowe, EGA, HEGA, VGA
- karty rozszerzenia pamięci
- kontrolery
- dyski twarde typu Winchester 20 MB, 40 MB, 60 MB, 80 MB

— systemy wielodostępne i lokalne sieci mikrokomputerowe:
- MULTI-LINK
- D-LINK
- XENIX

— materiały eksploatacyjne:
- dyskietki 5.25" MD2-D
- dyskietki 5.25" MD2-HD
- dyskietki 3" CF2
- dyskietki 3.5" MF 2DD
- taśmy barwiące do wszystkich typów drukarek STAR i NEC

Termin realizacji zamówień **natychmiast** po złożeniu zamówienia. **Bezpłatnie:** szkolenia, kursy, zestawy oprogramowania narzędziowego i użytkowego — przy dostarczeniu kompletnych systemów.

Proponujemy również na wszelkiego rodzaju mikrokomputery programy wspomagające zarządzanie przedsiębiorstwem:

— system finansowo-księgowo-kosztowy
— system zbytu i zaopatrzenia
— system technicznego przygotowania produkcji
— system materiałowy
— system kadrowy i kadrowo-płacowy (również dla pracowników akordowych)

Wymienione systemy pracują w wersjach sieciowych i wielodostępnych.

Wszelkich informacji udzielamy codziennie (oprócz sobót) w siedzibie Zakładu w Łodzi przy Al. Kościuszki 101, tel. 36-51-00, w godzinach 8—16.

EO/1005/87

ZDZISŁAW PŁOSKI
Instytut Informatyki
Uniwersytet Wrocławski

Jeden z wielu, choć może być T'woim jedynym!

CHIWRITER — procesor tekstu (I)

W niniejszym artykule scharakteryzowano właściwości i możliwości zastosowań programu Chiwriter, procesora tekstu dla mikrokomputerów typu IBM PC. Prócz informacji o obsłudze Chiwritera, przytoczono wybór ogólnych opinii o oprogramowaniu służącym przetwarzaniu tekstów i kilka sądów na temat cech dobrego oprogramowania powszechnego użytku dla mikrokomputerów osobistych.

Informacje pochodzą z oryginalnej instrukcji obsługi Chiwritera [1] (wersji dostępnej nieodpłatnie) oraz z wypowiedzi zawartych w pracy zbiorowej zatytułowanej Whole Earth Software Catalog, WESC [2].

TEMAT DLA SOCJOPSYCHOLOGA

Według jednych, procesory tekstu stanowią 80% realnych zastosowań komputerów osobistych, według innych — „tylko" 60%, przystańmy zatem na siedem dziesiątych. W takich warunkach warto posłuchać, co mówią na temat zjawiska przetwarzania tekstów doświadczeni, zauroczeni, a może tylko „zainteresowani" użytkownicy [2].

Stewart Brand (szef WESC): Programy przetwarzania tekstów są tym dla pisania, czym dla rachowania — kalkulatory.

Justin Kaplan (biograf): To jest (przetwarzanie tekstów) pociągające i zabawne, różne od maszynopisania niczym latanie od psiego chodu.

Steven Levy (redakcja WESC): Porównuję używanie procesorów tekstu do życia z drugą osobą. Zaczynasz z wielkim entuzjazmem i aprobatą, wszystko jest wtedy cudowne. Potem zauważasz, że inne (procesory tekstu) obiecują więcej. Większą różnorodność, przyjaźniejszy styl. Powstaje pytanie: czy warto zrywać związki, w które zainwestowałeś miesiące, dla jakiejś nowej sztuczki z przestawianiem słów, funkcji indeksującej, której użyjesz być może dwa razy, czy możliwości rozdzielania ekranu. Wybór procesora tekstu jest poważną życiową decyzją i nikt nie powinien (w kategoriach czasu, pieniędzy lub emocji) jej lekceważyć.

Alfred Lee: Odniosłem wrażenie, że jestem w swoistym transie. Kiedy moja żona zagadnęła mnie, co sądzę o kupnie lampy, nie mogłem pojąć o co chodzi — wyobcowany z jej świata, dopóki nie wstałem i nie odwróciłem wzroku od ekranu.

Stewart Brand: Pisanie jest tak bardzo osobistym zajęciem, że ludzie zaczynają się identyfikować ze swoimi programami redagowania tekstów i nie znoszą żadnych obiektywnych ocen wystawianych tym programom. Ludzie w większości używają pierwszego procesora tekstu, którego się nauczyli. Jest on wrodzonym językiem ich palców, a wszystkie pliki muszą dochowywać przysięgi wierności wobec jego reguł.

Anonim: Choć w interesach trudno dać pierwszeństwo zakupowi komputera do przetwarzania tekstów, zastosowanie to może usprawiedliwić całą komputerową inwestycję.

Wobec powszechnej fascynacji programami przetwarzania tekstów, nagminnego ich użycia i obserwowanej pogoni za coraz to nowymi odmianami tych programów, odnośnie opisanego dalej programu Chiwriter można dać Czytelnikowi jedno zapewnienie: aby móc go używać skutecznie, nie trzeba poświęcać miesięcy — wystarczy kilka godzin treningu (nawet jedna!). I gdyby przyszło z nim się rozstać, poczucie utraconych zabiegów nie będzie doskwierać zbyt mocno.

CHIWRITER — CZYLI TO, CO WIDZISZ, JEST TYM, CO OTRZYMASZ

Operujący wieloma krojami pisma system redagowania tekstów Chiwriter jest produktem amerykańskiej firmy Horstman Software Design, pochodzącym z lat 1985—1986 (dwie wersje). Należy do ergonomicznej rodziny edytorów tekstu realizujących tytułowe hasło: *jakim widzisz tekst na ekranie, takim go otrzymasz na drukarce!* Znaczy to, że tekst oglądany na ekranie jest wolny od jakichkolwiek znaków sterujących. Jest to zgodne z akceptowaną obecnie tendencją, której przeciwieństwem są systemy uwarunkowane wcześniejszymi technologiami, na przykład, niezwykle rozbudowany (około 6 MB) system TeX.

Firma Horstman Software Design nie zabrania niekontrolowanego rozpowszechniania swojego edytora. Stosuje tu ciekawą politykę zezwolenia na kopie pirackie (które sama wytwarza), rozpowszechniane obok handlowych. Jedyną różnicą jest pojawiający się w kopiach pirackich, co 5000 wierszy, znak handlowy firmy.

WYMAGANIA SPRZĘTOWE I ZGODNOŚĆ Z INNYM OPROGRAMOWANIEM

Chiwriter może pracować na mikrokomputerach zgodnych z IBM PC, wyposażonych w którąś z kart graficznych, drukarkę, co najmniej jeden dysk elastyczny i minimum 256 KB pamięci operacyjnej.

Wśród wielu drukarek, z którymi może współpracować ten program, można wyróżnić tak popularne u nas typy jak Epson i Gemini. Spośród kart graficznych można polecić kartę Hercules, która umożliwia oglądanie większej (niż karta CGA) liczby wierszy jednocześnie. Chiwriter jest oprogramowaniem niezbyt drogim, koszt typowej konfiguracji nie przekracza sumy 150 dolarów (dysk „piracki" kosztuje około 10 dolarów).

System Chiwriter pracuje pod kontrolą systemu operacyjnego DOS, poczynając od wersji DOS 1.1. Pliki Chiwritera są oznaczane standardowo typem CHI (rozszerzenie nazwy umieszczane po kropce w identyfikatorze pliku) i są plikami ASCII, przy czym każda dwuznakowa sekwencja rozpoczynająca się od ukośnika ma specjalne znaczenie dla interpretacji tekstu. Jest możliwe przeczytanie za pomocą Chiwritera zwykłego pliku ASCII i poddanie go dalszej obróbce (na przykład w celu uzupełnienia znaków diakrytycznych w polskich literach). Chiwriter może też, po małych modyfikacjach plików, przetwarzać pliki Wordstara oraz współpracować z anglojęzycznym korektorem ortograficznym MicroSpell.

OBSŁUGA PROGRAMU — UWAGI OGÓLNE

Wielką zaletą Chiwritera jest jego prosta obsługa, a zwłaszcza znikomy wkład pracy potrzebny do opanowania programu na tyle, by móc wykonywać użyteczne prace edytorskie. W oryginale system jest skonfigurowany dla drukarki Epson i karty graficznej IBM. Aby wymienić moduł obsługi drukarki, należy wydać makropolecenie CWCONFIG (któremu odpowiada program załączony na dysku) i wybrać odpowiedni typ drukarki. Pełną gamę sterowników drukarek obsługuje wersja handlowa. Przy innej karcie graficznej (np. karcie Hercules), należy dokonać zmian w pliku parametrów instalacyjnych Chiwritera.

Aby rozpocząć pracę, należy wydać polecenie:

CW [nazwa_pliku [. CHI]]

(nawiasy zaznaczają nieobowiązkowe części polecenia). Domyślnie obiera się dysk o symbolu A:. Można to zmienić pisząc pełny identyfikator pliku w parametrze. Jeśli rozpoczyna się pracę nad nowym dokumentem, pisze się tylko nazwę polecenia:

CW

Do systemowych plików Chiwritera jest dołączony krótki samouczek. Przestudiowanie go i wykonanie zawartych w nim ćwiczeń można gorąco polecić jako pełnoprawną przepustkę do użytkowania programu. Samouczek wprowadza się do pamięci poleceniem:

CW TUTORIAL

po czym, używając zwykłych klawiszy funkcyjnych z klawiatury IBM PC (strzałki, PGUP, PGDN) można go przeglądać w przód i wstecz. W następnym punkcie artykułu przytoczono informacje zawarte w pliku **TUTORIAL** uzupełnione o pewne dodatkowe spostrzeżenia.

Czynności związane z redagowaniem wyraża się za pomocą operacji klawiszowych (także — definiowanych przez użytkownika). Pozostała obsługa, np. drukowanie, składowanie czy strojenie systemu, odbywa się — jak przystało na typowy program usługowy PC — za pomocą hierarchicznie pogrupowanych poleceń tworzących wykazy (menu), co eliminuje potrzebę pamiętania ich nazw.

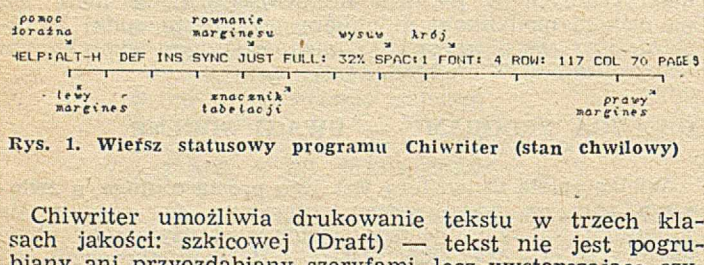
Szczególna funkcja przypada klawiszowi ESC. Jego użycie powoduje zmianę kontekstu stosowną do aktualnej sytuacji, na przykład przejście od trybu redagowania tekstu do wyświetlania głównego menu, powrót z menu szczegółowego do menu głównego itp.

Polecenia biorą nazwy od początkowych liter słów charakteryzujących ich działanie. W wersji Chiwriter 2 można też przemieszczać się po drzewie poleceń za pomocą kursora.

Funkcje specjalne uzyskuje się przez jednoczesne naciśnięcie klawisza CTRL albo klawisza ALT i któregoś z klawiszy literowych. Na przykład CTRL-F powoduje sformowanie bieżącego akapitu według aktualnych parametrów szerokości wiersza i odstępu między wierszami. W każdej chwili wykaz operacji klawiszowych można przywołać na ekran za pomocą polecenia ALT-H. Przy synchronicznym naciskaniu dwu klawiszy najpierw należy wcisnąć i przytrzymać klawisz sterujący, a później klawisz literowy, przy czym jest bez znaczenia czy wybrana litera będzie mała, czy wielka. Bardziej rozbudowane czynności, na przykład makrodefiniowanie, są sterowane konwersacyjnie.

Zmianę kroju pisma (10 zestawów w Chiwriterze 1, 20 — w wersji Chiwriter 2) uzyskuje się za pomocą klawiszy funkcyjnych klawiatury IBM PC, przy czym jednokrotne użycie klawisza oznacza zmianę doraźną, a dwukrotne — trwałą.

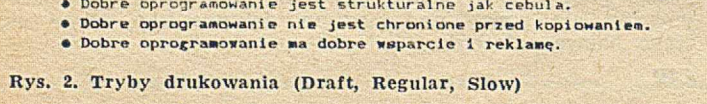
W każdej chwili użytkownik jest zorientowany co do podstawowych parametrów redagowania: szerokości wiersza, pozycji redagowanego znaku na stronie, kroju pisma, wielkości wysuwu itp. Stosowne informacje są zawsze wyświetlane na ekranie podczas redagowania (rys. 1).



Rys. 1. Wiersz statusowy programu Chiwriter (stan chwilowy)

Chiwriter umożliwia drukowanie tekstu w trzech klasach jakości: szkicowej (Draft) — tekst nie jest pogrubiany ani przyozdabiany szeryfami, lecz wystarczająco czytelny, normalnej (Regular) — można ją zalecić przy ważniejszej korespondencji — i podwyższonej (Slow), która w istocie powoduje znaczne spowolnienie wyprowadzania tekstu (przydatna, gdy mamy nienajświeższą taśmę barwiącą w drukarce). Różnice w trybach drukowania zilustrowano na rys. 2.

Chiwriter umożliwia wybór aktualnego skorowidza (kartoteki) bez konieczności schodzenia na poziom systemu operacyjnego. Możliwe jest przechowywanie i odzyskiwanie plików z różnych skorowidzów, a także wyprowadzanie zawartości skorowidzów (nazwy plików są wyświetlane w kolejności alfabetycznej).



Rys. 2. Tryby drukowania (Draft, Regular, Slow)

Istnieją operacje umożliwiające redagowanie systematyczne: odszukiwanie określonych fragmentów tekstu i ich automatyczne zastępowanie. Można też budować makrodefinicje, tj. opatrywać własnymi nazwami dowolne ciągi znaków (dokładniej, kodowaniu takiemu podlega ciąg naciśnięć dowolnych klawiszy). W ten sposób można dokonywać zarówno podręcznych skrótów nazwowych (makrodefinicje w klasycznym rozumieniu), a także — co ważniejsze — zamykać w formie nowych operacji ciągi innych operacji.

Szereg elementów programu podlega (na życzenie) strojeniu przez użytkownika. W tej liczbie, można decydować o postaci nagłówków, pojemności strony, częstości automatycznych składowań, sposobie drukowania (drukowanie fragmentów pliku), obecności bądź nieobecności pewnych krojów pisma.

Obfitość krojów pisma, ich definiowalność (polecenie FD) oraz definiowalność operacji klawiszowych, technika rozszerzania bądź zawężania wysokości wierszy (liczby „ząbków" w terminologii klasycznej maszyny do pisania) — powodują, że Chiwriter może być z powodzeniem użyty do przygotowywania tekstów zawierających wielopoziomowo indeksowane oznaczenia, wzory matematyczne, a nawet tabele, rubryki itp.

SZCZEGÓŁY PROGRAMU CHIWRITER — WYCIĄG Z PLIKU TUTORIAL

W tym punkcie podano większość informacji dotyczących edytora Chiwriter 1 (w wersji z 1985 roku). Na opisanie modelu z roku 1986, konsekwentnie rozwijającego koncepcje pierwszego wydania, nie zdecydowano się z powodu przedstawionego w zakończeniu artykułu. Większość zamieszczonych tu informacji zainteresowany użytkownik uzyska studiując plik TUTORIAL.

Operowanie kursorem

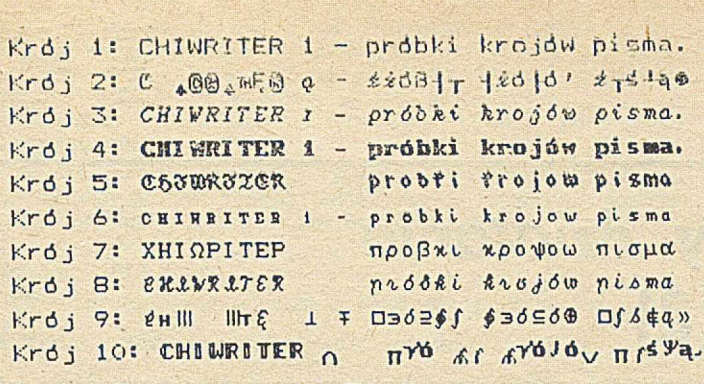
Manewrowanie kursorem odbywa się w Chiwriterze w sposób typowy. Używa się tu klawiatury dodatkowej IBM PC (cyfry, strzałki). Tekst można przeglądać znak po znaku, przeskakując słowa (CTRL-LEFT, CTRL-RIGHT), wiersze (HOME, END, CTRL-PGUP, CTRL-PGDN) lub strony (PGUP, PGDN). Przejście do początku lub końca tekstu umożliwiają operacje CTRL-HOME i — odpowiednio — CTRL-END. Ponadto, jest możliwa tabulacja (TAB, SHIFT-TAB) i przeskok do następnej lub poprzedniej litery, powodowany przez naciśnięcie klawisza PLUS lub MINUS z dodatkowego rejestru klawiszy po prawej stronie i docelowego znaku. Na przykład, PLUS-z przemieści kursor w przód do najbliższej litery „z", a sekwencja MINUS-F3-C — przesunie kursor wstecz do litery C. W praktyce może to się wydać nawet nadmiarem możliwości — użytkownik szybko nabiera własnego stylu w operowaniu kursorem. Wszelako pozostawanie wyłącznie przy strzałkach jest na pewno samoograniczeniem się powodującym stratę czasu.

Wstawianie i kasowanie tekstu

Chiwriter robi właściwy użytek z klawisza INS jak również z klawisza ← (ang. backspace) i DEL. Działanie klawisza INS — tryb wstawiania lub tryb nakładania — jest przemienne. Tryb wstawiania znaków jest sygnalizowany napisem INS w wierszu wyświetlającym status edytora.

Kroje pisma

W tej mierze trudno o jednolitą ocenę. Dla piszącego te słowa dziesięć krojów Chiwritera 1, to i tak o połowę za dużo (rys. 3). Wyrafinowany matematyk być może odczuje niedosyt oznaczeń i przy 20 krojach Chiwritera 2. Z podstawowych krojów można wymienić liternictwo zbliżone do antykwy (klawisz funkcyjny F1: ABCDEF...), znaki specjalne — w szczególności litery polskie (klawisz F2: ĄŁŃćśŹ) i wiele innych. Litery polskie występują także w innych krojach pisma. Krój F3, to kursywa: *ABCdef*, F4 — pismo pogrubione: **ABCdef**, F6 — pismo niskie. Do tego dochodzą: greka — F7, gotyk — F5, pisanka — F8 i dwa kroje ze znakami bądź fragmentami znaków ma-



Rys. 3. Próbki pisma programu Chiwriter

tematycznych: — math I — F9 i math II — F10. Numer kroju pisma uwidacznia na ekranie rubryka FONT. Jest możliwe definiowanie nowych krojów pisma, służy do tego oddzielny program (Font Designer), wchodzący w skład wyposażenia. Zmianę kroju pisma uzyskuje się przez wciśnięcie odpowiedniego klawisza funkcyjnego. Dołączone na dysku pliki z krojami w układzie klawiaturowym i „alfabetycznym" oraz możliwość doraźnego wyświetlenia ich na ekranie (*Fm* ALT-H) pozwalają na dobrą orientację. Mistrzostwo osiąga się drogą ćwiczeń.

Budowa wiersza

Przy pisaniu nie trzeba zważać na zmiany wiersza. Chiwriter nie dopuści do podzielenia słowa zmianą wiersza. Z chwilą osiągnięcia aktualnego prawego marginesu niedokończony wyraz jest automatycznie przenoszony do nowego wiersza, przy czym następuje (lub nie — por. formatowanie) automatyczne dosuwanie tekstu do prawego marginesu w zapewnionym wierszu (rozsunięcie słów) i dodanie przez program własnej, „wirtualnej", nowej linii.

Zmiany wiersza wprowadzone przez użytkownika (naciśnięcie klawisza ENTER, rzadziej oznaczanego jako RETURN) są granicami akapitów (ang. paragraphs) — por. formatowanie.

Wiersz może się rozszerzać w pionie bądź skracać, co umożliwia „wielopiętrowe" indeksowanie wzorów. Stosowne operacje to: CTRL-A — dodanie nowego rządka w wierszu, CTRL-Z — usunięcie rządka. Aczkolwiek Chiwriter automatycznie rozszerza wiersze w pionie w razie zaistniałej potrzeby, gdy użytkownik przemieścił kursor i kontynuuje pisanie, z pewnych względów (jak np. niepatrzne przejście w strefę sąsiedniego wiersza) lepiej jest przy pisaniu wzorów w rodzaju:

$$\int_a^b f(x)\,dx$$

najpierw dodać pewien zapas rządków (CTRL-A ...), a następnie wpisać wzór i na koniec ewentualnie usunąć zbędne światło (CTRL-Z ...) — por. też z operacją ALT-*a* (ustawianie wysuwu).

W rozszerzonych w pionie wierszach działanie klawisza DEL (lub ←) odnosi się do całej kolumny. Może to wywołać niepożądane skutki, na przykład przy próbie usunięcia liter „a" i „b" z powyższego wzoru. Aby tego uniknąć i nie skasować wraz z tymi literami znaku całki, należy wprowadzić program w tryb niesynchronicznego traktowania kolumny znaków w wierszu za pomocą operacji CTRL-S. Operacja ta na przemian włącza i wyłącza działanie synchronicznego kasowania lub wstawiania znaków w rządkach wiersza; aktualny status jest zawsze wyświetlany na ekranie — napis SYNC.

Marginesy, wysuw i tabulacja

Aktualna lokalizacja i szerokość wiersza jest zaznaczana na ekranie odcinkiem poprowadzonym wzdłuż wiersza za statusami operacji. Do zmiany marginesów służą operacje CTRL-L i CTRL-R (odpowiednio — lewy i prawy margines). Margines jest ustawiany wzdłuż bieżącego położenia kursora w wierszu. Maksymalna szerokość wiersza wynosi 77 znaków.

Wysuw, to jest liczba rządków z których składa się — mierząc w pionie — każdy wiersz, jest ustawiany za pomocą funkcji ALT-0 ... ALT-9, które w ograniczonym przedziale automatyzują rozsuwanie wiersza za pomocą operacji CTRL-A. Funkcja ALT-0 odpowiada wysuwowi minimalnemu (dwa „ząbki" na maszynie do pisania), ALT-1 — trzem „ząbkom" itd. Wartość aktualna wysuwu jest stale wyświetlana na ekranie w rubryce o nazwie SPAC.

Operacja CTRL-T powoduje naprzemienne ustawianie i likwidowanie tabulatora w kolumnach wskazywanych przez kursor.

LITERATURA

[1] Horstman Software Design: Chiwriter Multifont Word Processing System — Tutorial. Ann Arbor (MI), 1985
[2] Whole Earth Software Catalog. 2nd edition. San Francisco (CA), 1986.

Ceny ogłoszeń

Od 1 lipca 1987 r. obowiązują następujące ceny materiałów reklamowych publikowanych na łamach INFORMATYKI:

Ogłoszenia

— ogłoszenia czarno-białe, artykuły reklamowe i informacje naukowo-techniczne (biuletyny) zależnie od objętości: cała strona — 50 tys., 3/4 str. — 45 tys., 2/3 str. — 40 tys., 1/2 str. — 35 tys., 1/3 str. — 30 tys., 1/4 str. — 25 tys., 1/8 str. — 20 tys., poniżej 1/8 str. — 200 zł za 1 cm²,
— ogłoszenia drobne (zależnie od liczby słów) jedno słowo — 50 zł

Dodatki do ceny podstawowej:

— za każdy dodatkowy kolor + 30%,
— za każdy specjalny kolor (nie wynikający z podstawowych kolorów) + 30%
— za pełny kolor (grafika wielobarwna, zdjęcia kolorowe) + 120%
— za zamieszczenie ogłoszenia na I lub IV stronie okładki + 100%
— za zamieszczenie ogłoszenia na II i III stronie okładki + 50%

Zniżki

dotyczą ogłoszeń — całkowitych powtórzeń
— za ogłoszenia 3—5-krotne — 10%
— za ogłoszenia 6—10-krotne — 20%
— za ogłoszenia 11-krotne i powyżej — 30%
— za artykuły i wkładki reklamowe wykonane przez zleceniodawcę — 40%
— za biuletyny i bloki reklamowe — 60%

W uzasadnionych wypadkach stosuje się zniżki specjalne dla ogłoszeń nie będących powtórzeniami — za zgodą Dyrektora — Naczelnego Redaktora Wydawnictwa NOT SIGMA.
Ponadto Biuro Ogłoszeń świadczy usługi w zakresie wykonywania zdjęć czarno-białych i barwnych oraz nadbitek wkładek reklamowych.

Ceny wkładek

— wkładka 2 str. o formacie A4
nakład do 500 egz. 20 tys. zł
nakład 500—1000 egz. 35 tys. zł

— wkładka 4 str. o formacie A4
nakład 500 egz. 40 tys. zł
nakład 500—1000 egz. 70 tys. zł

Przy wkładkach o nakładzie powyżej 1000 egz. stosuje się wielokrotność powyższych cen, a dodatki za kolory oblicza się jak dla ogłoszeń.

Ogłoszenia przyjmowane są przez:
Dział Ogłoszeń i Reklamy WCIKT NOT SIGMA
ul. Świętojerska 5/7, 00-236 Warszawa
adres do korespondencji: skrytka pocztowa 1004, 00-950 Warszawa
telefony: 31-93-65 lub 31-22-21 w. 196 i 291

Uprzejmie informujemy Czytelników, że egzemplarze INFORMATYKI — bieżące i archiwalne — można kupić nie tylko w kioskach Ruchu, Klubie NOT SIGMY, Zakładzie Kolportażu i Dziale Handlowym (szczegóły podano w WARUNKACH PRENUMERATY), ale również
w lokalu naszej redakcji
ul. Mickiewicza 18 m. 17 w Warszawie, tel. 39-14-34
oraz
w specjalistycznej księgarni PP „Domu Książki"
ul. Mokotowska 51/53 w Warszawie, tel. 28-16-14
Zapraszamy wszystkich zainteresowanych.

STANISŁAWA OSSOWSKA
Gdańsk

Wordstar 3.30 — zasady działania i sposób użytkowania (3)

Trzecia, ostatnia część artykułu jest poświęcona omówieniu poleceń zawartych w menu dla ekranu i menu sterowania wydrukiem, a także tzw. poleceń z kropką.

MENU DLA EKRANU

Menu dla ekranu (ang. onscreen menu) umożliwia wprowadzanie zmian do wygenerowanej podczas instalowania odmiany użytkowej Wordstara, czego efekty są widoczne na ekranie. Zawiera pięć grup poleceń (rys. 1).

```
^T      A:NAUKA  PAGE 1 LINE 1 COL 01                    INSERT ON
                  ( ( (     P R I N T   M E N U     ) ) )
------ Special  Effects -------   ! -Printing  Changes-  !  -Other  Menus-
(begin and end) !  (one time each)  ! A Alternate pitch   ! (from Main only)
B Bold D Double ! H Overprint char  ! N Standard pitch    !^J Help  ^K Block
 S Underscore   ! O Non-break space ! C Printing pause    !^Q Quick ^P Print
 X Strikeout    ! F Phantom space   ! Y Other ribbon color!^O Onscreen
 V Subscript    ! G Phantom rubout  !  --User  Patches--  !Space Bar returns
 T Superscript  ! RET Overprint line ! 0(1) 0(2) E(3) R(4) !you to Main Menu.
L----!----!----!----!----!----!----!----!----!----!--------R
```

Rys. 1. Polecenia zawarte w menu dla ekranu

● **Marginesy i tabulacje (ang. margins and tabs):**

L — ustawianie lewego marginesu (zwykle pierwszą kolumnę);
R — ustawianie prawego marginesu (zwykle 65 kolumna);
X — zwalnianie ustawionych marginesów i umożliwienie wpisywania tekstu poza nimi (działa na zasadzie przełącznika);
I — ustawianie kolejnych pozycji tabulacji zmiennej przez wprowadzanie znaku # i numeru kolumny dla tabulacji dziesiętnej, bądź tylko numeru kolumny dla tabulacji normalnej;
N — usuwanie niepotrzebnych pozycji tabulacji (jednocześnie wszystkich lub jednej);
G — przesunięcie całego akapitu do najbliższej pozycji tabulacji; działanie tego polecenia kończy się z chwilą naciśnięcia klawisza RETURN;
F — ustawianie zarówno obu marginesów jak i pozycji tabulacji.

● **Funkcje dla wierszy (ang. line functions):**

C — centrowanie tekstu między aktualnie ustawionymi marginesami;
S — ustawianie odstępu między wierszami (zwykle wartość 1, co jest sygnalizowane w wierszu statusu brakiem informacji o odstępie między wierszami).

● **Przełączniki (ang. toggles):**

W — włączanie lub wyłączanie automatycznego przenoszenia wyrazów do nowego wiersza (zwykle całe wyrazy nie mieszczące się w wierszu ograniczonym marginesami są przenoszone do nowego wiersza automatycznie; wyłączenie automatycznego przenoszenia wyrazów wymaga naciśnięcia klawisza RETURN w celu przejścia do nowego wiersza;
T — włączanie lub wyłączanie wyświetlania wiersza marginesu i tabulacji (parametry ustalone dla tego wiersza działają niezależnie od ustawienia tego przełącznika).

● **Pozostałe przełączniki (ang. more toggles):**

J — włączanie lub wyłączanie automatycznego rozsuwania wyrazów w celu wyrównania do prawego marginesu;
V — ustawianie trybu stałej lub zmiennej tabulacji; tabulatory ustawione dla zmiennej tabulacji są zaznaczone w wierszu marginesu i tabulacji za pomocą wykrzykników (standardowo co 5 kolumn) i można je zmieniać poleceniem **^OI**; obraz tabulacji stałej — standardowo co 8 kolumn — nie jest wyświetlany;
H — włączanie lub wyłączanie samouczka przy przenoszeniu długich słów;
E — interpretowanie znaków przeniesienia „—"; znaki przeniesienia przy włączonym przełączniku są drukowane tylko wówczas, gdy znajdują się jako ostatnie znaki tekstowe w wierszu; wyłączenie przełącznika powoduje drukowanie wszystkich znaków przeniesienia wprowadzonych z klawiatury;
D — włączanie lub wyłączanie wyświetlania znaków sterujących wydrukiem;
P — włączanie lub wyłączanie wiersza wskazującego koniec strony (jest to wiersz będący linią przerywaną zakończony literą P w skrajnej prawej kolumnie).

MENU STEROWANIA WYDRUKIEM

Menu sterowania wydrukiem (ang. print menu) zawiera pewien repertuar poleceń, pozwalających otrzymywać efekty specjalne dopiero podczas drukowania. W trakcie redagowania tekstu, na ekranie pojawiają się dodatkowe znaki nie stanowiące treści pliku, lecz pełniące funkcje sterujące pracą drukarki, wskutek umieszczenia tych znaków w tekście, obserwuje się przekraczanie marginesów na ekranie o liczbę wprowadzonych znaków sterujących wydrukiem — nie są one jednak wliczane do liczebności znaków w wierszu.

```
^O      A:NAUKA  PAGE 1 LINE 1 COL 01                    INSERT ON
                 ( ( (  O N S C R E E N   M E N U  ) ) )
-Margins & Tabs-  ! -Line  Functions-  !  --More Toggles--  !  -Other  Menus-
L Set left margin !C Center text       !J Justify   now ON  ! (from Main only)
R Set right margin!S Set line spacing  !V Vari-Tabs now ON  !^J Help  ^K Block
 X Release margins !                   !H Hyph-help now ON  !^Q Quick ^P Print
I Set  N Clear tab!   ---Toggles---    !E Soft hyph now OFF !^O Onscreen
G Paragraph tab   !W Wrd wrap now ON   !D Prnt disp now ON  !Space Bar returns
F Ruler from line !T Rlr line now ON   !P Pge break now ON  !you to Main Menu.
L----!----!----!----!----!----!----!----!----!----!--------R
```

Rys. 2. Polecenia menu sterowania wydrukiem

Należy pamiętać, że wiele efektów uzyskiwanych podczas drukowania zależy od możliwości drukarki oraz poprawnego zainstalowania Wordstara uwzględniającego te możliwości.

● **Efekty specjalne (ang. special effects):**

B — wytłuszczenie fragmentu tekstu znajdującego się między znacznikami **^B**;
D — dwukrotne drukowanie tekstu znajdującego się między znacznikami **^D**;
S — podkreślanie fragmentu tekstu znajdującego się między znacznikami **^S**;
X — przekreślanie fragmentów tekstu między znacznikami **^X** przez nadrukowanie na tym tekście znaków „—";
V — wydrukowanie fragmentu tekstu między znacznikami **^V** przesunięte zwykle o połowę odstępu między wierszami poniżej pozostałej części wiersza;
T — drukowanie fragmentu tekstu między znacznikami **^T** przesunięte zwykle o połowę odstępu między wierszami powyżej pozostałej części wiersza.

Powyższe efekty specjalne osiąga się dla dowolnego fragmentu pliku między ustawionymi znacznikami, które mogą być widoczne na ekranie. W poleceniach podanych poniżej efekty specjalne dotyczą pojedynczych znaków lub jednego wiersza:
H — drukowanie znaku następującego po tym poleceniu na znaku poprzednim (może to być wykorzystane do drukowania znaków narodowych, jeśli nie ma wygodniejszych rozwiązań);

O — wprowadzanie spacji do drukowanego tekstu, powodującej nierozdzielanie wyrazów nią połączonych (na ekranie zamiast spacji wyświetla się ^O); wyrazy połączone tym poleceniem są traktowane jak jeden wyraz i nie są rozdzielane ani w trakcie redagowania, ani ponownego formatowania;

F — drukowanie znaku specjalnego w miejscu spacji, np. graficznego obrazu spacji; postać znaku zależy od dostępnego repertuaru znaków oraz od instalacji;

G — drukowanie znaku graficznego odpowiadającego klawiszowi DEL;

RET — drukowanie wiersza na wierszu poprzednim.

● **Zmiana rodzaju druku (ang. printing changes):**

A — inicjowanie innego rodzaju drukowania, np. większej szerokości, czy innego pochylenia czcionki; efekty drukowania alternatywnego wymagają specjalnych właściwości drukarek oraz prawidłowej instalacji Wordstara;

N — przywracanie drukowania czcionką podstawową;

C — zatrzymanie drukowania w miejscu, w którym znajduje się ten znak (ma zastosowanie wówczas, gdy jest konieczna ręczna obsługa drukarki); w linii statusu pojawia się informacja PRINTED PAUSED;

Y — zmiana koloru taśmy barwiącej do wystąpienia kolejnego polecenia **^Y**, które przywraca kolor poprzedni.

● **Inne efekty drukowania (ang. user patches):**

Q(1), W(2), E(3), R(4) — sterowanie dodatkowymi możliwościami drukarki, nie uwzględnionymi w powyższych poleceniach, zależnie od rodzaju drukarki (konkretyzowane w trakcie instalowania Wordstara).

POLECENIA Z KROPKĄ

Z poleceniami sterującymi wydrukiem jest związana inna grupa poleceń, tzw. polecenia z kropką (DOT COMMANDS). Polecenia z kropką sterują tworzeniem nagłówków, stopek, ustalaniem formatu strony, jeśli zainstalowana odmiana użytkowa nie jest odpowiednia (wartość przyjętych parametrów można wyświetlić poleceniem **^Jd**) wysuwaniem papieru do nowej strony, numerowaniem stron, generowaniem wydruku itp. Są one wprowadzane do tekstu jako specjalne, oddzielne wiersze złożone z kropki w pierwszej kolumnie oraz dwuliterowego kodu. Polecenia te są wyświetlane na ekranie, lecz pomijane zwykle w trakcie drukowania.

● **Ustawianie nagłówka i stopki**

.HE tekst — definiowanie nagłówka, który będzie drukowany aż do następnego polecenia .HE (gdy brak tekstu, przyjmuje się spacje); jeśli polecenia nie użyto, to nagłówek nie będzie drukowany;

.FO tekst — definiowanie stopki, która będzie umieszczana na każdej stronie aż do wystąpienia następnego polecenia .FO; zastępuje ona numerację stron, jeśli nie polecono inaczej.

● **Znaki specjalne w nagłówku (.HE) i stopce (.FO)**

— wskazanie miejsca w nagłówku bądź stopce, w którym ma być drukowany aktualny numer strony;

| (ukośnik) — interpretowanie następnego znaku literalnie, a nie jako znaku specjalnego;

^K — umożliwienie drukowania numerów stron na końcach tych stron; wprowadzane poleceniem **^PK**.

● **Przygotowanie do drukowania (nowa strona):**

.PA — bezwarunkowe przesunięcie papieru na drukarce do nowej strony;

.CP n — warunkowe przesunięcie papieru na drukarce do nowej strony, w zależności od liczby wolnych wierszy na stronie.

● **Sterowanie numerowaniem stron:**

.OP — pomijanie drukowania numerów stron poczynając od bieżącej strony;

.PN — drukowanie numerów stron poczynając od strony bieżącej;

.PN n — wprowadzenie numeru strony **n** i rozpoczęcie drukowania kolejnych numerów stron;

.PC n — określanie numeru kolumny, w której ma być drukowany numer strony (zwykle przyjmuje się środkową kolumnę tekstu na stronie).

● **Formatowanie strony:**

.PO n — ustalanie lewego odstępu marginesu na stronie innego niż przyjęty w zainstalowanej odmianie użytkowej; **n** oznacza numer kolumny, od której ma być drukowany tekst (zwykle przyjmuje się 8);

.PL n — długość strony wyrażona w całkowitej liczbie wierszy na stronie (zwykle przyjmuje się 66);

.MT n — ustawianie górnego marginesu wyrażanego liczbą pustych wierszy od fizycznego początku strony do początku tekstu (zwykle przyjmuje się 3);

.MB n — ustawianie dolnego marginesu wyrażanego liczbą pustych wierszy między ostatnim wierszem poprzedniej strony a pierwszym wierszem następnej strony (zwykle przyjmuje się 8);

.HM n — ustawienie marginesu między nagłówkiem a początkiem tekstu (zwykle przyjmuje się dwa puste wiersze);

.FM n — ustawianie marginesu między końcem tekstu a stopką (zwykle przyjmuje się dwa puste wiersze).

● **Polecenia tylko dla drukarek o specjalnych możliwościach:**

.LH n — ustawianie wysokości wiersza;

.CW n — ustawianie parametrów dla innego rodzaju druku; polecenie równoważne **^PA**;

.SR n — ustawianie wartości parametru dla drukowania poniżej bądź powyżej wiersza.

● **Inne polecenia:**

..tekst — umieszczanie w pliku komentarza, który nie będzie drukowany;

.UJ OFF/ON — wyłączanie lub włączanie automatycznego rozsuwania wyrazów w celu wyrównania do prawego marginesu; przed pisaniem tablic celowe jest stosowanie polecenia .UJ OFF, a po jego zakończeniu — .UJ ON.

Pozostałe polecenia z kropką są związane z generatorem wydruków MailMerge i nie będą przedstawione.

* * *

Nie wszystkie polecenia Wordstara są udostępniane w menu, np. PL inicjujące przejście do nowej strony. Pełne poznanie możliwości tego programu wymaga sięgnięcia do bardziej szczegółowych opracowań.

Do słabszych stron Wordstara należy zaliczyć: brak możliwości graficznych, brak możliwości definiowania własnych znaków, brak rozwiązań automatycznie przemieszczających tekst stosownie do innego planu, inna postać tekstu na ekranie niż postać wydruku. Nie przesłaniają one jednak jego niewątpliwych zalet.

Język awk

dokończenie ze str. 19

Język awk znalazł zastosowania — zdaniem jego twórców — dość nieoczekiwane. Implementowano za jego pomocą bazy danych, a nawet rozmaite kompilatory i asemblery. Zasadniczym jego zastosowaniem pozostaje jednak wyszukiwanie informacji, operowanie danymi, produkowanie sprawozdań, statystyk — wszystko zgodnie z prostą i skuteczną filozofią: wiedzieć (wyłożyć) "co zrobić?" i "z czym zrobić?".

LITERATURA

[1] TekstCSK. Podręcznik operatora. Computer Studio Kajkowscy, Gdynia, 1984.
[2] Wordstar. Installation Manual. Version 3.0. MicroPro International Corporation, San Rafael, CA, 1982
[3] Wordstar. Podręcznik użytkownika. Zakład Informatyki Przemysłu Okrętowego, Gdańsk, 1985
[4] Wordstar. Training Guide, Sanyo, 1984.

ANNA SYFERT
ANDRZEJ RAZNOWIECKI
(oprac.)
Warszawa

Struktura systemu operacyjnego PC-DOS (3)

Wywołań systemowych dokonuje się przez przekazanie kodu funkcji do procedury systemowej. Ponieważ część funkcji jest zgodna z systemem CP/M, a część z systemem Xenix, w systemie PC-DOS w zasadzie mogą wykonywać się programy zarówno systemu CP/M jak i Xenixa. Jednakże, w rzeczywistości całkowita zgodność między tymi systemami nie została zapewniona. Z tego względu, w poniższym opisie przyjęto dość nietypową konwencję omawiania wywołań. Mianowicie, zamiast omawiania ich w kolejności lub podziału logicznego według rodzajów funkcji, zastosowano podział określony przez zgodność z systemem operacyjnym. Ponieważ dokładny opis wszystkich wywołań i ich parametrów wykraczałby poza ramy objętościowe artykułu, ograniczono się jedynie do podania krótkiego przeglądu możliwości, jakie dostaje do dyspozycji programista.

Jako pierwsze przedstawiono znane funkcje charakterystyczne dla systemu operacyjnego CP/M. Funkcje typu CP/M są wywoływane przez rozkaz krótkiego wywołania CALL do adresu 05H wewnątrz prefiksu segmentu programu, z kodem funkcji zawartym w rejestrze CL. Przyporządkowanie rejestrów jest praktycznie identyczne z przyporządkowanie dla CP/M-86, jednak również użytkownik systemu CP/M-80 znajduje tu znane sobie właściwości. Dwa inne mechanizmy wywoływania funkcji (długie wywołanie do adresu 50H i przerwanie INT 21H) nie są zgodne z CP/M i pierwszą wersją PC-DOS; wyboru funkcji dla tych wywołań dokonuje się przez rejestr AH.

FUNKCJE ZNAKOWEGO WEJŚCIA-WYJŚCIA

Funkcja 01H — wczytanie znaku z klawiatury.
Oczekuje na wprowadzenie znaku z urządzenia CON, czyta wprowadzony znak, wyprowadza go na ekran monitora (echo) i wraca do wywołującego programu. Jeśli w momencie wywołania w 128-bajtowym buforze klawiatury znajdują się już znaki, to one zostaną najpierw przekazane do wywołującego programu. Funkcja może zostać przerwana przez CTRL-C (wykonanie przerwania 23H).
Parametr wejściowy:
AH-01H
Parametr wyjściowy:
AL — wczytany znak

Funkcja 02H — wyprowadzenie znaku na ekran.
Wyprowadza znak na urządzenie CON. Funkcję można przerwać przez CTRL-C (INT 23H).
Parametry wejściowe:
AH-02H
DL — wyprowadzany znak

Funkcja 03H — wczytanie znaku przez sprzęg szeregowy.
Czeka na wprowadzenie znaku przez urządzenie AUX lub COM1, z możliwością przerwania przez CTRL-C (INT 23H). Jeśli w buforze znajdują się już znaki, to one zostaną najpierw przekazane do programu wywołującego.
Parametr wejściowy:
AH-03H
Parametr wyjściowy:
AL — znak ASCII wczytany z AUX

Funkcja 04H — wyprowadzenie znaku przez sprzęg szeregowy.
Wyprowadza znak na urządzenie AUX lub COM1. Funkcja ta może zostać przerwana przez CTRL-C (INT 23H).
Parametry wejściowe:
AH-04H
DL — znak ASCII wyprowadzony przez AUX

Funkcja 05H — wyprowadzenie znaku na drukarkę.
Wyprowadza znak na drukarkę PRN, z możliwością przerwania przez CTRL-C (INT 23H)
Parametry wejściowe:
AH-05H
DL — znak ASCII wyprowadzany na drukarkę

Funkcja 06H — bezpośrednie wyprowadzanie z konsoli i wprowadzanie na konsolę.
Wprowadza lub wyprowadza znak przy użyciu urządzenia CON. Funkcja ta może zostać przerwana przez CTRL-C; CTRL-C może zostać wczytane jako znak, jeśli została wyłączona kontrola CTRL-C za pomocą funkcji 33H.
Parametry wejściowe:
AH-06H
DL — znak wyprowadzany (FFH oznacza wprowadzanie znaku przez rejestr AL).
Parametr wyjściowy:
AL — wczytany znak (gotowość znaku jest sygnalizowana przez bit flagowy zera, jeśli wywołania dokonano za pomocą DL=FFH).

Funkcja 07H — wczytanie znaku z klawiatury bez echa.
Oczekuje na znak z klawiatury (jeśli jeszcze nie ma znaku) i przekazuje go do programu wywołującego. Znak nie jest wyprowadzany na ekran monitora (brak echa). Funkcja ta może zostać przerwana przez CTRL-C (wprowadzenie CTRL-C przekazuje 03H; CTRL-C może zostać wczytany jako znak, jeśli zostanie wyłączona kontrola CTRL-C za pomocą funkcji 33H).
Parametr wejściowy:
AH-07H
Parametr wyjściowy:
AL — znak wprowadzony w kodzie ASCII (jeśli AL=0, to następne wywołanie wprowadza znak specjalny).

Funkcja 08H — wczytanie znaku z klawiatury bez echa.
Wykonuje się jak funkcja 01H, lecz bez echa.
Parametr wejściowy:
AH-08H
Parametr wyjściowy:
AL — wczytany znak ASCII

Funkcja 09H — wyprowadzenie napisu.
Wyprowadza ciąg znaków zakończony znakiem "$" ("$" nie zostanie wyprowadzony, a w przypadku napisu zawierającego znak "$" należy zastosować specjalną obsługę).
Parametry wejściowe:
AH-09H
DS:DX — adres wyprowadzanego napisu.

Funkcja 0AH — wczytanie napisu.
Wprowadza znaki z urządzenia CON (z echem) tak długo, aż napotka znak CR lub zostanie osiągnięta liczba znaków określona przez program wywołujący. Udostępnia wczytany napis oraz jego długość programowi wywołującemu. Funkcja ta może zostać przerwana za pomocą CTRL-C (INT 23H). Przy wprowadzaniu większej liczby znaków niż wyspecyfikowane zostaną one zignorowane, a użytkownik będzie poinformowany o pełnym buforze sygnałem dźwiękowym BEL (07H). Powrót do programu wywołującego dokona się dopiero po wprowadzeniu znaku CR (0DH). Znak CR zostanie wpisany do bufora, jednak nie będzie uwzględniony w liczbie wprowadzonych znaków. Podczas wprowadzania można używać klawiszy BACKSPACE i DELETE w celu korygowania wprowadzanych znaków. Wielkość obszaru przeznaczonego na wprowadzany napis musi być większa o 2 od maksymalnej liczby wprowadzanych znaków (tabela).
Parametry wejściowe:
AH-0AH

DS:DX — adres obszaru przeznaczonego na wprowadzony napis (p. tabela)

Budowa obszaru przeznaczonego na napis

Wyrównanie	Zawartość
0	Maksymalna liczba wczytywanych znaków (włącznie z CR); musi zostać ustawiona przed wywołaniem funkcji
1	Liczba rzeczywiście wprowadzonych znaków (CR nie jest liczony)
2...	Bufor na znaki

Funkcja 0BH — sprawdzenie stanu klawiatury.
Stwierdza, czy w buforze klawiatury znajdują się znaki. Jeśli znakiem jest CTRL-C, to wykona się INT 23H.
Parametr wejściowy:
AH-0BH
Parametr wyjściowy:
AL-FFH, jeśli są wprowadzane znaki
AL-00H, jeśli nie ma żadnych wprowadzonych znaków

Funkcja 0CH — zerowanie bufora klawiatury i wykonanie wprowadzania.
Bufor klawiatury zostaje wyzerowany i następuje wykonanie funkcji o kodzie zawartym w AL (jeśli nie jest to dopuszczalny kod funkcji, to bufor zostanie tylko wyzerowany).
Parametry wejściowe:
AH-0CH
AL-01H, 06H, 07H, 08H lub 0AH
DL — znak wyprowadzany
Parametr wyjściowy:
AL — znak wprowadzany (00H oznacza, że bufor konsoli został wyzerowany, lecz nie została wywołana żadna funkcja).

```
;       NAME    TEST_PRINTER
; PROGRAM WPROWADZA NA DRUKARKE KOLEJNE ZNAKI ASCII W WIERSZACH
; O DLUGOSCI 80 ZNAKOW. PIERWSZY WIERSZ ZACZYNA SIE OD SPACJI (20H)'
; NASTEPNY OD '!' (21H) ITD. WPROWADZANIE ZNAKOW ZOSTANIE PRZERWANE
; W WYNIKU PRZYCISNIECIA DOWOLNEGO KLAWISZA .
;
STACK   SEGMENT STACK
        DW      20 DUP(?)
STACK   ENDS
CODE    SEGMENT 'CODE'
ASSUME  CS:CODE, DS:CODE, ES:NOTHING, SS:STACK
CR      EQU     0DH     ; ZDEFINIOWANIE ZNAKU RETURN
DOS     EQU     21H     ; PRZERWANIE DLA WYWOLANIA FUNKCJI
KEY_STATUS EQU  0BH     ; FUNKCJA DOS: czy w buforze
                        ; ZNAJDUJE SIE JAKIS ZNAK?
LF      EQU     0AH     ;ZDEFINIOWANIE ZNAKU LINEFEED
OUT_STRING EQU  09H     ; FUNKCJA DOS: wyprowadzenie ciagu
                        ; znakow
PRINT_CHARACTER EQU 05H ; FUNKCJA DOS: drukowanie znaku
RESET_KEY EQU   0CH     ; FUNKCJA DOS: wyczyszczenie
                        ; bufora konsoli
SPACE   EQU     20H     ; ZDEFINIOWANIE ZNAKU SPACE
TERMINATE EQU   4CH     ; FUNKCJA DOS: zakonczenie programu
START_TEXT DB   ' ZACZYNA SIE TEST DRUKARKI ',CR,LF,LF,'$'
END_TEXT DB     ' TEST DRUKARKI ZOSTAL PRZERWANY ',CR,LF,LF,'$'
START:  MOV     AX,CS
        MOV     DS,AX   ; ustawienie segmentu danych
        MOV     AH,RESET_KEY ; wyczyszczenie bufora konsoli
        MOV     AL,0FFH ; nie wykonywanie zadnej z funkcji WE
        INT     DOS
        MOV     DX,OFFSET START_TEXT
                ; ustawienie adresu komunikatu poczatkowego
        MOV     AH,OUT_STRING
                ; FUNKCJA 09H
        INT     DOS     ; wykonanie funkcji
INIT:   MOV     DL,SPACE
AGAIN:  MOV     DL,BL   ; pierwszy znak dla wyprowadzenia
        MOV     CX,80   ; CX=80(050H), znaki/wiersz
        MOV     AH,PRINT_CHARACTER
                ; FUNKCJA 05H
LOOP:   INT     DOS     ; drukowanie znaku
        INC     DL      ; nastepny znak
        CMP     DL,7FH
        JNZ     DALEJ   ; nastepny znak <> 7FH
        MOV     DL,SPACE
                ; stluminie znakow 00H - 1FH
        OR      DL,DL   ; wyzerowanie wskaznika ZERO
DALEJ:  LOOPNZ  LOOP    ; wyprowadzenie 80 znakow
        MOV     DL,CR   ; wyprowadzenie znaku RETURN
                ; koniec wiersza
        INT     DOS
        MOV     DL,LF
        INT     DOS     ; wyprowadzenie znaku LINEFEED
        MOV     AH,KEY_STATUS
                ; okreslenie statusu klawiatury
        INT     DOS
        OR      AL,AL   ; czy jest wprowadzony znak z klawia-
                ; tury
        JZ      PRINT   ; nie, nastepny wiersz
        MOV     DX,OFFSET END_TEXT
                ; adres komunikatu koncowego
        MOV     AH,OUT_STRING
                ; FUNKCJA: wyprowadzenie ciagu znakow
        INT     DOS
        MOV     AH,TERMINATE
                ; zakonczenie programu
        INT     DOS
PRINT:  INC     BL      ; pierwszy znak nastepnego wiersza
        CMP     BL,7FH  ; zmiana
        JNZ     AGAIN   ; 7FH nie zostalo jeszcze osiagniete
        JMP     INIT    ; powtorny poczatek od znaku SPACE
CODE    ENDS
        END     START
```

Wydruk 1. Przykład zastosowania niektórych funkcji od 01H do 0CH

Na wydruku 1 przedstawiono zastosowanie kilku opisanych funkcji. Wszystkie przykłady w tym artykule zostały sprawdzone pod nadzorem systemu PC-DOS 2.11. Do asemblacji i łączenia użyto makroasemblera MASM V3.0 i linkera Microsoft V02.

FUNKCJE ZARZĄDZANIA PLIKAMI

Poniżej omówiono operacje na plikach dostosowane do systemu CP/M. Należy pamiętać o tym, że funkcje zarządzania plikami tego rodzaju nie opierają się na hierarchicznej strukturze skorowidzów systemu PC-DOS. Korzystają natomiast z bloku FCB. Przy wywołaniu funkcji jest jej przekazywany początkowy adres bloku FCB.

Funkcja 0DH — zerowanie dysku.
Zapewnia, że wszystkie bufory użyte do przesłania danych na nośnik zostaną przepisane, a następnie zwolnione. Oprócz tego, wszystkie bufory zawierające dane dla otwartych plików zostają wyczyszczone, przy czym skorowidz nie zostanie zaktualizowany (zmienione pliki muszą zostać jawnie zamknięte). Funkcji tej używa się w celu uzyskania zdefiniowanego stanu systemu plików i powinno się stosować ją, na przykład, w procedurze obsługi przerwania CTRL-C. Przy wymianie dyskietki funkcja ta jest konieczna tylko wtedy, jeśli nie wszystkie pliki, które uległy zmianom, zostały zamknięte.
Parametr wejściowy:
AH-0DH

Funkcja 0EH — wybór dysku.
Wybiera dysk, dzięki czemu można odwołać się do wszystkich plików znajdujących się na odnośnym dysku, bez potrzeby podawania specyfikacji dysku.
Parametry wejściowe:
AH-0EH
DL — numer dysku (A=0, B=1 itd.)
Parametr wyjściowy:
AL — liczba dysków

Funkcja 0FH — otwarcie pliku.
Jeśli plik zostanie znaleziony, to do już otwartego bloku FCB zostaną wpisane pewne informacje (wyrównanie 00H — numer dysku, 0CH-0DH — chwilowy numer bloku równy 0, 0EH-0FH — długość bloku danych równa 128 bajtów, 10H-13H — wielkość pliku, 14H-17H — data i godzina ostatniego zapisu pliku). Przed odwołaniem się do pliku, należy w danym przypadku ustawić wyrównanie 0EH-0FH na rzeczywistą długość bloku danych. Przy dostępie sekwencyjnym należy ustawić wyrównanie 20H (aktualny rekord danych), przy dostępie bezpośrednim należy odpowiednio opisać wyrównanie 21H-24H. Jeśli plik nie zostanie znaleziony, to należy go utworzyć za pomocą funkcji 16H. Jeśli plik, który należy otworzyć, ma szczególne atrybuty, to należy użyć odpowiedniego, rozszerzonego bloku FCB, gdyż w przeciwnym przypadku plik nie zostanie znaleziony.
Parametry wejściowe:
AH-0FH
DS:DX — adres nieotwartego FCB
Parametr wyjściowy:
AL-00H, jeśli plik istnieje
AL-FFH, jeśli plik nie został znaleziony

Funkcja 10H — zamknięcie pliku
Otwarty plik zostanie zamknięty, a skorowidz zostanie zaktualizowany.
Parametry wejściowe:
AH-10H
DS:DX — adres otwartego bloku FCB
Parametr wyjściowy:
AL-00H, jeśli plik istnieje
AL-FFH, jeśli plik nie został znaleziony

Funkcja 11H — odszukanie pierwszego pliku o zadanej nazwie.

Przeszukuje skorowidz pod względem wystąpienia pierwszego rozszerzenia danego pliku (mogą tu zostać użyte znaki zapytania). Jeśli zostanie znaleziony pierwszy wpis danego pliku do skorowidza, to nieotwarty blok FCB przekazywany w wywołaniu zostaje skopiowany do bufora dysku (DTA). Pierwotny blok FCB zostaje zmodyfikowany w celu dalszego przeszukiwania przy użyciu funkcji 12H i dlatego nie może już zostać zmieniony. Jeśli poszukuje się plików o specjalnych atrybutach, to należy użyć rozszerzonego bloku FCB.

Parametry wejściowe:
AH-11H
DS:DX — adres nieotwartego bloku FCB
Parametr wyjściowy:
AL-00H, jeśli plik został znaleziony
AL-FFH, jeśli plik nie został znaleziony

Funkcja 12H — odszukanie następnego pliku o zadanej nazwie.

Przeszukuje skorowidz pod względem wystąpienia następnego wpisu pliku o danej nazwie. Przedtem należy użyć funkcji 11H w celu znalezienia pierwszego wpisu. Obowiązują tutaj wszystkie uwagi jak dla funkcji 11H.

Parametry wejściowe:
AH-12H
DS:DX — adres nieotwartego bloku FCB, użytego przez funkcję 11H
Parametr wyjściowy:
AL-00H, jeśli plik został znaleziony
AL-FFH, jeśli plik nie został znaleziony

Funkcja 13H — usunięcie pliku.

Usuwa jeden lub kilka plików. Przy podaniu nazwy pliku można użyć znaku "?". Pliki, które mają specjalne atrybuty, mogą zostać usunięte wyłącznie przy użyciu rozszerzonego bloku FCB.

Parametry wejściowe:
AH-13H
DS:DX — adres nieotwartego bloku FCB
Parametr wyjściowy:
AL-00H, jeśli plik został znaleziony
AL-FFH, jeśli plik nie został znaleziony

Funkcja 14H — odczyt sekwencyjny.

Przed wykonaniem tej funkcji należy ustawić w bloku FCB wyrównanie 0CH-0DH (aktualny rekord danych) oraz 20H (aktualny rekord danych w bloku danych). Wtedy funkcja wczytuje do bufora dyskowego rekord danych wybrany przez parametry w FCB i zmienia odpowiednio oba wyżej wymienione parametry.

Parametry wejściowe:
AH-14H
DS:DX — adres otwartego FCB
Parametr wyjściowy:
AL-00H, jeśli proces odczytu zakończył się bez błędu
AL-01H, jeśli osiągnięto koniec pliku, ale nie zostały przeczytane żadne dane
AL-02H, jeśli proces odczytu nie został przeprowadzony, ponieważ bufor nie był dostatecznie duży do wczytania rekordu danych
AL-03H, jeśli został osiągnięty koniec pliku przy odczycie rekordu; rekord jest uzupełniany do pełnej długości znakami 00H (koniec danych rzeczywiście wczytanych jest oznaczony przez CTRL-Z. 1AH lub musi zostać obliczony na podstawie długości pliku — wyrównanie 10H-13H w bloku FCB).

Funkcja 15H — zapis sekwencyjny.

Przed wykonaniem tej funkcji należy ustawić w bloku FCB wyrównanie 0CH-0DH (aktualny blok danych) oraz 20H (aktualny rekord danych w bloku danych). Funkcja zapisuje wtedy rekord danych z bufora dyskowego systemu PC-DOS oraz zmienia odpowiednio wymienione parametry. Choć bez użycia rozszerzonego FCB nie można skasować pliku z atrybutem tylko do odczytu, to przy użyciu zwykłego FCB jest możliwe otwarcie pliku z ochroną zapisu oraz zapisanie go.

Parametry wejściowe:
AH-15H
DS:DX — adres otwartego FCB
Parametr wyjściowy:
AL-00H, jeśli proces zapisu został zakończony bez błędów
AL-01H, jeśli nośnik danych jest zapełniony (proces zapisu nie został przeprowadzony)
AL-02H, jeśli proces zapisu nie został przeprowadzony, ponieważ w buforze nie ma dostatecznie dużego, wolnego obszaru do zapisania rekordu.

Funkcja 16H — utworzenie pliku.

Utworzony plik o długości 0, otwierany następnie przez funkcję 0FH. Jeśli tworzony plik już istnieje, to zostanie on usunięty, a na jego miejsce powstanie plik pusty o tej samej nazwie. Jeśli przy tworzeniu pliku jest używany rozszerzony blok FCB, to nowy plik otrzymuje atrybuty wyspecyfikowane w wyrównaniu -1H bloku FCB.

Parametry wejściowe:
AH-16H
DS:DX — adres nieotwartego bloku FCB
Parametr wyjściowy:
AL-00H, jeśli plik został założony
AL-FFH, jeśli plik nie mógł być założony, ponieważ w skorowidzu nie ma już miejsca.

Funkcja 17H — przemianowanie pliku.

Przed wywołaniem funkcji należy wygenerować blok FCB, zawierający pierwotną nazwę pliku (wyrównanie 01H-0BH), nową nazwę pliku (wyrównanie 11H-1BH) oraz numer dysku (wyrównanie 00H). Obie nazwy plików mogą zawierać znaki "?". W celu przemianowania plików szczególnie chronionych należy użyć rozszerzonego bloku FCB. Użycie dwóch identycznych nazw plików prowadzi do błędu.

Parametry wejściowe:
AH-17H
DS:DX — adres specjalnego bloku FCB
Parametr wyjściowy:
AL-00H, jeśli dokonano przemianowania
AL-FFH, jeśli nie znaleziono pliku, który miał być przemianowany lub już istnieje plik o takiej nazwie.

Funkcja 19H — sprawdzenie wybranego dysku.

Przy użyciu tej funkcji można stwierdzić, który dysk został ostatnio wybrany za pomocą funkcji 0EH.

Parametr wejściowy:
AH-18H
Parametr wyjściowy:
AL — wybrany dysk (0=B, 1=B itd.)

Funkcja 1AH — zmiana adresu bufora dla dyskowych operacji wejścia-wyjścia (DTA).

Zmienia DTA (ang. disk transfer address) domyślnie określony przez wyrównanie 80H obszaru PSP (maksymalna długość — 128 bajtów). Należy zwrócić uwagę na to, aby bufor dysku nie przekroczył granic segmentu (jeśli DTA zostanie ustawiony na 2500H:FFF0H, to bufor dysku ma maksymalnie 16 bajtów).

Parametry wejściowe:
AH-1AH
DS:DX — nowy adres początku bufora dysku

Funkcja 1BH — odczytanie informacji o wybranym dysku.

Dostarcza pożytecznych informacji o aktualnie wybranym dysku.

Parametr wejściowy:
AH-1BH
Parametry wyjściowe:
DS:DX — adres bajtu identyfikacyjnego tablicy FAT (ang. file allocation table [1])
CX — długość sektora w bajtach (standardowo — 512 bajtów)
AL — liczba sektorów na jednostkę alokacji (najmniejszą jednostkę, na jakie dzieli się obszar dysku); w przypadku dyskietek zapisanych dwustronnie, odpowiada dwom sektorom, a na dyskach stałych o pojemności 10 MB, odpowiada ośmiu sektorom
DX — liczba jednostek alokacji, jakie są do dyspozycji na danym dysku

Funkcja 1CH — odczytanie informacji o dowolnym dysku.

Dostarcza identycznych informacji o dowolnym dysku jak w przypadku funkcji 1BH.

Parametry wejściowe:
AH-1CH
DL — numer dysku (0 — bieżący dysk, 1=A, 2=B itd.)
Parametry wyjściowe:
jak dla funkcji 1BH

Funkcja 21H — odczyt pliku o dostępie swobodnym.

Przed wykonaniem tej funkcji należy ustawić w bloku FCB wyrównanie 21H-24H (numer zaadresowanego rekordu danych w odniesieniu do początku pliku). Funkcja dopasowuje w FCB odpowiednio długość rekordu danych

(wyrównanie 0EH-0FH), wskaźnik aktualnego bloku danych (wyrównanie 0CH-0DH) i wczytuje do bufora dysku (DTA) rekord danych wyspecyfikowany przez te parametry.

```
;
;Niniejszy program wyszukuje w wybranym katalogu zbiory
;z rozszerzeniem .BAK i nastepnie usuwa je.
;Przetworzenie Disk Transfer Address nie jest konieczne i
;sluzy wylacznie celom dydaktycznym.
;Zbiory z ochrona zapisu wprawdzie zostanie znaleziony,
;jednak nie zostanie usuniety.
;
CR              EQU     0DH     ;Carriage Return
DELETE          EQU     13H     ;Funkcja DOS: usuwanie zbioru
DOS             EQU     21H     ;Wywolanie funkcji DOS
LF              EQU     0AH     ;Linefeed
OUT_STRING      EQU     09H     ;Funkcja DOS: wyprowadzenie
                                ;ciagu znakow
SEARCH_FIRST    EQU     11H     ;Funkcja DOS: szukanie pierw-
                                ;szego extentu w katalogu
SEARCH_NEXT     EQU     12H     ;Funkcja DOS: szukanie nastep-
                                ;nego extentu
SET_DTA         EQU     1AH     ;Funkcja DOS: ustawienie nowe-
                                ;go DTA
TERMINATE       EQU     4CH     ;Funkcja DOS: zakonczenie pro-
                                ;gramu
;
STACK   SEGMENT STACK
                DW      20 DUP(?)
STACK   ENDS
CODE            SEGMENT 'CODE'
ASSUME          CS:CODE, DS:CODE, ES:CODE, SS:STACK
DELETED         DB      '.BAK zostal usuniety',CR,LF,'$'
NOT_DELETED     DB      '.BAK nie zostal usuniety',CR,LF,'$'
;
DTA             DB      128 DUP(?)
                                ; rezerwacja pamieci
                                ; na bufor dysku
FCB             DB      0,'????????BAK'
                                ; nieotwarty FCB ze
                                ; wzorcem dla szukania
                DB      25 DUP(?)
                                ; uzupelnic FCB do pel-
                                ; nej dlugosci (37 bajt)
;
;
START:          MOV     AX,CODE         ; ustawienie segmentu
                                        ; danych
                MOV     DS,AX
                MOV     ES,AX           ; ustawienie segmentu
                                        ; dodatkowego
                MOV     DX,OFFSET DTA   ; ustawienie nowego DTA
                MOV     AH,SET_DTA      ; ustawienie DTA zystemu
                                        ; DOS
                INT     DOS             ; wywolanie funkcji
                MOV     AH,SEARCH_FIRST ; DOS: szukanie pierwsze-
                                        ; go extentu
LOOP:           MOV     DX,OFFSET FCB   ; ustawienie adresu FCB
                                        ; dla funkcji szukania
                INT     DOS             ; wywolanie funkcji
                OR      AL,AL           ; czy zostal znaleziony
                                        ; extent?
                JNZ     FINISHED        ; *.BAK nie istnieje,
                                        ; koniec programu
                CALL    ERASE           ; usuniecie zbioru
                MOV     AH,SEARCH_NEXT  ; DOS: szukanie nastepne-
                                        ; go extentu
                JMP     LOOP            ; koniec programu lub
                                        ; usuniecie zbioru
FINISHED:
                MOV     AH,TERMINATE    ; DOS: zakonczenie pro-
                                        ; gramu
                INT     DOS             ; wywolanie funkcji
;
ERASE           PROC
                MOV     DX,OFFSET DTA   ; ustawienie adresu na
                                        ; FCB dla wywolanie
                                        ; funkcji usuwania
                MOV     AH,DELETE       ;DOS: funkcja usuwania
                INT     DOS             ; wywolanie funkcji
                PUSH    AX              ; ratowanie statusu
                                        ; bledu
                MOV     AL,'
                MOV     DI,OFFSET DTA + 1
                MOV     CX,9
                MOV     DX,DI           ; ustawienie adresu dla
                                        ; wyprowadzenia
                REPNE   SCASB           ; szukanie pierwszej
                                        ; spacji w nazwie zbioru
                MOV     BYTE PTR [DI-1],'$'
                                        ; ogranicznik dla wyprowa-
                                        ; dzenia ciagu znakow
                MOV     AH,OUT_STRING   ; DOS: wyprowadzenie ciagu
                                        ; znakow ( nazwy zbioru)
                INT     DOS             ; wywolanie funkcji
                POP     AX              ; badanie statusu bledu
                OR      AL,AL           ; czy zbior zostal usunie-
                                        ; ty?
                JZ      SUCCESS         ; tak
                MOV     DX,OFFSET NOT_DELETED
                                        ; wyprowadzenie komunikatu
                                        ; bledu
                JMP     COMMON_OUT      ; skok w celu wyprowadzenia
                                        ; ciagu znakow
SUCCESS:
                MOV     DX,OFFSET DELETED
                                        ; wyprowadzenie komunikatu
                                        ; o wykonaniu programu
COMMON_OUT:
                MOV     AH,OUT_STRING   ; DOS: wyprowadzenie ciagu
                                        ; znakow
                INT     DOS             ; wywolanie funkcji
                RET
ERASE           ENDP
CODE            ENDS
                END     START
```

Wydruk 2. NAME DELETE_BAK

Parametry wejściowe:
AH-21H
DS:DX — adres otwartego bloku FCB
Parametr wyjściowy:
AL-00H, jeśli proces czytania został zakończony bez błędu
AL-01H, jeśli został osiągnięty koniec pliku (EOF), a nie zostały przeczytane żadne dane
AL-02H, jeśli proces czytania nie został przeprowadzony, ponieważ w buforze nie ma wolnego miejsca dla rekordu danych
AL-03H, jeśli przy czytaniu rekordu został osiągnięty koniec pliku; rekord jest uzupełniany przez wypełnienie znakiem 00H do oczekiwanej długości (koniec rzeczywiście wczytanych danych jest oznaczony przez CTRL-Z lub musi zostać obliczony na podstawie długości pliku — wyrównanie 01H-13H w FCB)

Funkcja 22H — zapis pliku o dostępie swobodnym.
Tak jak przy odczycie w dostępie swobodnym należy naj-

```
;Niniejszy program listuje zawartosc zbioru ASCII
;procedura wyprowadzania uzywa funkcji 9
;i w przypadku wystapienia znaku '$' wewnatrz wpro-
;wadzanego ciagu znakow dokonuje specjalnej obslugi
;
CHAR_OUT        EQU     02H     ;funkcja DOS: wyprowadzenie znaku
CLOSE           EQU     10H     ;funkcja DOS:zamkniecie zbioru
CONTROL_Z       EQU     1AH     ;znak zakonczenia zbioru
CURRENT_RECORD  EQU     20H     ;offset aktualnego rekordu w FCB
DOS             EQU     21H     ;wywolanie funkcji DOS
DTA_SMALL       EQU     02H     ;blad czytania: bufor dysku za maly
EOF             EQU     01H     ;wskaznik konca zbioru przy czytaniu
ORIGINAL_FCB    EQU     5CH     ;adres pierwszego FCB w PSP
OPEN            EQU     0FH     ;funkcja DOS: otwarcie zbioru
STRING_OUT      EQU     09H     ;funkcja DOS:wyprowadzenie ciagu
                                ;znakow
PARTIAL_REC     EQU     03H     ;ostatni przeczytany rekord nie jest
                                ;pelny
READ            EQU     14H     ;funkcja DOS: czytanie sekwencyjne
SET_DTA         EQU     1AH     ;funkcja DOS: ustawienie nowego DTA
TERMINATE       EQU     4CH     ;funkcja DOS: zakonczenie programu
;
STACK           SEGMENT STACK
                DB      20 DUP(?)
STACK           ENDS
;
DATA            SEGMENT 'DATA'
BUFFER          DB      128 DUP(?)
                                ;bufor dla dysku
FCB             DB      37 DUP(?)
                                ;kopia Original-FCB1 przy wykonaniu
DATA.           ENDS
;
CODE            SEGMENT 'CODE'
;
ASSUME          CS:CODE, DS:DATA, ES:DATA, SS:STACK
;
START:          MOV     AX,DATA
                MOV     ES,AX   ;inicjalizacja segmentu dodatkowego
                MOV     DI,OFFSET FCB
                                ;adres przeznaczenia zawartosci FCB
                MOV     SI,ORIGINAL_FCB
                                ;poczatek FCB1
                MOV     CX,12   ;dlugosc nieotwartego FCB
                REPNE   MOVSB   ;skopiowanie FCB
                MOV     DS,AX   ;inicjalizacja segmentu danych
                MOV     DX,OFFSET BUFFER
                                ;ustawienie adresu bufora dysku
                MOV     AH,SET_DTA
                                ;DOS: zalozenie nowego bufora dysku
                INT     DOS     ;wywolanie funkcji DOS
                MOV     DX,OFFSET FCB
                MOV     AH,OPEN ;DOS: otwarcie zbioru
                INT     DOS     ;wywolanie funkcji DOS
                MOV     BYTE PTR FCB[CURRENT_RECORD],0
                                ;inicjalizacja aktualnego rekordu w
                                ;FCB
LOOP:           MOV     DX,OFFSET FCB
                MOV     AH,READ ;DOS: czytanie sekwencyjne
                INT     DOS     ;wywolanie funkcji DOS
                MOV     DX,OFFSET BUFFER
                                ;ustawienie adresu ciagu znakow
                MOV     CX,LENGTH BUFFER
                                ;ustawienie dlugosci bufora
                CMP     AL,EOF  ;czy zostal osiagniety koniec zbioru?
                JZ      FINISHED
                                ;tak, koniec
                CMP     AL,DTA_SMALL
                                ;czy bufor dysku jest zbyt maly?
                JZ      FINISHED
                                ;tak,koniec
                CMP     AL,PARTIAL_REC
                                ;czy ostatni nie calkiem pelny
                                ;rekord danych?
                JNZ     OUTPUT  ;nie, wyprowadzenie
                MOV     DI,DX
                MOV     AL,CONTROL_Z
                REPNE   SCASB   ;odszukanie w ciagu znakow konca zbioru
                MOV     AX,LENGTH BUFFER - 1
                SUB     AX,CX   ;obliczenie dlugosci reszty ciagu zna-
                                ;kow
                MOV     CX,AX
                CALL    OUT_STRING
                                ;wyprowadzenie czesciowego ciagu znakow
                JMP     FINISHED;zakonczenie programu
OUTPUT:         CALL    OUT_STRING
                                ;ponowne wyprowadzenie ciagu znakow
                JMP     LOOP    ;nastepny rekord danych
FINISHED:
                MOV     DX,OFFSET FCB
                MOV     AH,CLOSE;DOS: zamkniecie zbioru
                INT     DOS     ;wywolanie funkcji
                MOV     AH,TERMINATE
                                ;DOS: zakonczenie programu
                INT     DOS     ;wywolanie funkcji
```

```
OUT_STRING      PROC
                CMP     CX,0    ;czy dlugosc ciagu znakow = 0?
                JZ      RETURN  ;powrotem do wywolujacego programu
                MOV     DI,DX
CONTI:          MOV     AL,'$'
                REPNE   SCASB   ;szukanie '$' w ciagu znakow
                CMP     CX,0    ;czy nie znaleziono?
                JNZ     OUT     ;został znaleziony '$', wyprowa-
                                ;dzic czesciowy ciag znakow
                MOV     BL,[DI-1]
                                ;reszta ciagu znakow nie zawiera
                                ;znaku '$' , ratowanie ostatniego
                                ;znaku w BL
                MOV     BYTE PTR [DI-1],'$'
                                ;zastapienie ostatniego znaku przez '$'
OUT:            MOV     AH,STRING_OUT
                                ;funkcja: wyprowadzenie ciagu znakow
                INT     DOS     ;wywolanie funkcji
                CMP     CX,0    ;wyprowadzenie ostatniego czesciowego
                                ;ciagu znakow
                JZ      LAST    ;tak,wyprowadzenie ostatniego znaku
                                ;w (BL)
                MOV     DL,'$'  ;wyprowadzenie '$'
                MOV     AH,CHAR_OUT
                                ;funkcja: wyprowadzenie znaku
                INT     DOS     ;wywolanie funkcji
                MOV     DX,DI   ;ustawienie adresu nastepnego czesciowe-
                                ;go ciagu znakow
                JMP     CONTI
LAST:           MOV     DL,BL   ;wyprowadzenie ostatniego znaku
                MOV     AH,CHAR_OUT
                                ;funkcja: wyprowadzenie znaku
                INT     DOS     ;wywolanie funkcji
RETURN:         RET
OUT_STRING      ENDP
;
CODE            ENDS
                END     START
```

Wydruk 3. NAME VIEW_FILE

pierw ustawić w bloku FCB względny numer zapisywanego rekordu danych. Funkcja dopasowuje numer aktualnego bloku danych (wyrównanie 0CH-0DH) oraz aktualnego rekordu w tym bloku danych (wyrównanie 20H) odpowiednio do wybranej długości bloku danych (wyrównanie 0EH-0FH) oraz zapisuje rekord danych o tych parametrach na nośniku (lub do wewnętrznego bufora dysku). Tak jak w przypadku zapisu o dostępie sekwencyjnym zapis do pliku z ochroną zapisu jest możliwy.
Parametry wejściowe:
AH-22H
DS:DX — adres otwartego bloku FCB
Parametr wyjściowy:
AL-00H, jeśli proces zapisu został zakończony bez błędu
AL-01H, jeśli nośnik danych jest w pełni zapisany; proces zapisu nie może zostać wykonany
AL-02H, jeśli proces zapisu nie może zostać przeprowadzony, ponieważ w buforze nie ma dostatecznie dużo miejsca dla rekordu.

Funkcja 23H — określenie wielkości pliku.
Przed wykonaniem funkcji należy podać długość bloku danych (wyrównanie 0EH-0FH), chociaż blok FCB nie jest otwarty (możliwe jest użycie znaku "?" w nazwie pliku). Przy użyciu rozszerzonego FCB możliwe jest obejście ochrony pliku. Funkcja dostarcza wielkość pierwszego pliku, noszącego nazwę odpowiadającą wzorcowi. Jest ona dostarczana jako wielokrotność wybranej długości bloku (wynik jest zawarty w wyrównaniu 21H-24H bloku FCB). Zatem może się zdarzyć, że iloczyn długości bloku oraz liczby bloków udostępnianej przez funkcję jest większy od rzeczywistej wielkości pliku. Przypadek ten występuje zawsze wtedy, gdy wielkość pliku nie jest podzielna bez reszty przez długość bloku.
Parametry wejściowe:
AH-22H
DS:DX — adres nieotwartego bloku FCB
Parametr wyjściowy:
AL-00H, jeśli plik został znaleziony
AL-FFH, jeśli plik nie został znaleziony

Funkcja 24H — ustawienie względnego dodatkowego numeru rekordu.
Numer względnego rekordu danych (w odniesieniu do początku pliku) dla dostępu swobodnego (wyrównanie 21H-24H) jest ustawiany w bloku FCB odpowiednio do długości bloku danych (wyrównanie 0EH-0FH) oraz aktualnego rekordu danych (wyrównanie 20H) w ten sposób, aby został zaadresowany ten sam rekord co przy dostępie sekwencyjnym.
Parametry wejściowe:
AH-24H
DS:DX — adres otwartego bloku FCB

Na wydruku 2 przedstawiono przykład obu funkcji poszukiwania oraz usuwania plików. Wydruk 3 ilustruje użycie funkcji wyprowadzania napisów oraz użycie niektórych funkcji zarządzania plikami.

SANOCKA FABRYKA AUTOBUSÓW

zatrudni natychmiast

chętnych do pracy na stanowiskach

PROGRAMISTÓW-

-technologów obróbki wiórowej

oraz

ELEKTRONIKÓW

do obsługi serwisowej obrabiarek sterowanych numerycznie.

Jesteśmy zainteresowani kandydatami z kilkuletnim stażem pracy w danej dziedzinie oraz absolwentami szkół wyższych ze znajomością języka angielskiego lub niemieckiego.

Informujemy o możliwości otrzymania w ciągu roku mieszkania rodzinnego w Przyzakładowej Spółdzielni Mieszkaniowej, a do tego czasu zakład oferuje zakwaterowanie na warunkach hotelowych w Domu Młodego Robotnika.

Warunki pracy i płacy do uzgodnienia osobiście lub telefonicznie **w dziale osobowym — telefon 50-468.**

EO/1099/87

JAN BIELECKI
Instytut Informatyki
Politechnika Warszawska

Język C — konstruowanie, interpretowanie i wyznaczanie wartości wyrażeń

Wśród wyrażeń języka C szczególną rolę odgrywają wyrażenia pierwotne i l-wyrażenia. W pierwszym przybliżeniu, definicje tych pojęć można wyrazić za pomocą schematów przedstawionych w tabelach 1 i 2. Mimo iż wiele prac na temat języka C ogranicza się do przytoczenia podobnych definicji, łatwo jest skonstruować przykłady programów błędnych, niesprzecznych z wymaganiami wynikającymi z wymienionych tabel.

Tabela 1. Definicja pojęcia wyrażenie-pierwotne

```
l-wyrażenie:
  identyfikator
  wyrażenie-pierwotne [ wyrażenie ]
  l-wyrażenie-pierwotne . identyfikator
  wyrażenie-pierwotne -> identyfikator
  * wyrażenie
  ( l-wyrażenie )
```

Tabela 2. Definicja pojęcia l-wyrażenie

```
wyrażenie-pierwotne:
  identyfikator
  literał
  ( wyrażenie )
  wyrażenie-pierwotne ( lista-wyrażeń )
  wyrażenie-pierwotne [ wyrażenie ]
  l-wyrażenie-pierwotne . identyfikator
  wyrażenie-pierwotne -> identyfikator
```

W celu uniknięcia błędów należy mieć na uwadze następujące reguły dodatkowe:

1. Nie jest l-wyrażeniem napis, który reprezentuje wskazanie obiektu albo pole bitowe struktury.

2. W napisie o postaci

wyrażenie-pierwotne (lista-wyrażeń)

wyrażenie-pierwotne powinno być wyrażeniem pierwotnym reprezentującym funkcję.

3. W napisie o postaci

wyrażenie-pierwotne [wyrażenie]

dokładnie jedno z wymienionych wyrażeń powinno być wyrażeniem reprezentującym wskazanie elementu tablicy.

4. W napisie o postaci

l-wyrażenie-pierwotne . identyfikator

l-wyrażenie-pierwotne powinno reprezentować strukturę albo unię, a identyfikator powinien być nazwą pola tej struktury albo unii.

5. W napisie o postaci

wyrażenie-pierwotne —> identyfikator

wyrażenie-pierwotne powinno reprezentować wskazanie struktury albo unii, a identyfikator powinien być nazwą pola tej struktury albo unii.

Po skonstruowaniu wyrażenia poprawnego istotne staje się określenie sposobu jego interpretowania. Zależy to od ustalonych w języku priorytetów i wiązań operatorów, przytoczonych w tabeli 3.

Tabela 3. Priorytety i wiązania operatorów

Priorytet	Wiązanie	Operator
15	lewe	() [] ->
14	prawe	! ~ ++ -- - (typ) * & sizeof
13	lewe	* / %
12	lewe	+ -
11	lewe	<< >>
10	lewe	< <= > >=
9	lewe	== !=
8	lewe	&
7	lewe	^
6	lewe	|
5	lewe	&&
4	lewe	||
3	prawe	?:
2	prawe	= += -= *= /= %= >>= <<= &= |= ^=
1	lewe	,

Operacje związane z operatorami o wyższym priorytecie są wykonywane w pierwszej kolejności. Jeśli pewnego argumentu operacji dotyczą dwa operatory o równym priorytecie, to kolejność wykonywania operacji jest określona przez wiązanie. Uwzględnienie priorytetu i wiązań operatorów umożliwia w wielu wypadkach wyeliminowanie z wyrażeń zbędnych nawiasów okrągłych. W szczególności, poprawne jest wyrażenie:

a —= b = —c++ + d — e

całkowicie pozbawione nawiasów okrągłych. W wyrażeniu tym argumentu c dotyczą dwa operatory o równym priorytecie: operator zmiany znaku (—) i operator postinkrementacji (++). Ponieważ wiązanie każdego z tych operatorów jest prawostronne, w pierwszej kolejności zostanie wykonana operacja postinkrementacji, a dopiero potem zmiana znaku. Z tego powodu rozpatrywane wyrażenie jest traktowane tak jakby miało postać:

a —= b = (—(c++)) + d — e

Ponieważ priorytety operatorów dodawania (+) i odejmowania (—) są wyższe od priorytetu operatora przypisania (=), można wyrazić to posługując się parą dodatkowych nawiasów

a —= b = ((—(c++)) + d — e)

Operacje dodawania (+) i odejmowania (—) występujące w obrębie zewnętrznych nawiasów mają równy priorytet.

Ponieważ operatory dodawania (+) i odejmowania (—) wiążą argumenty lewostronnie, rozpatrywane wyrażenie jest interpretowane tak jak wyrażenie

a —= b = (((—(c++)) + d) — e)

Dla odmiany operatory przypisania wiążą swoje argumenty prawostronnie, a więc ostatecznie można wywnioskować, że pierwotne wyrażenie

a —= b = —c++ + d — e

jest interpretowane tak jak wyrażenie

(a —= (b = (((—(c++)) + d) — e)))

```
#include <stdio.h>

struct{
  int Fix;
} Str[] = { 100,200 },
  *Ptr  = Str;

main()
{
  printf("%d", ++ (Ptr ++) -> Fix );
  printf("%d", -(-- Ptr -> Fix)   );
}
```

Wydruk 1. Program poprawny

Na wydruku 1 przedstawiono pewien program poprawny, którego wykonanie powoduje wyprowadzenie liczb 101 i —199. Gdyby z tego programu usunięto wewnętrzne nawiasy, a więc gdyby przekształcono go na postać przedstawioną na wydruku 2, to byłby on niepoprawny z dwóch powodów:

- w pierwszej instrukcji printf przed operatorem „—" nie wystąpiłoby wyrażenie pierwotne,
- w drugiej instrukcji printf przed operatorem „++" nie wystąpiłoby l-wyrażenie.

```
#include <stdio.h>

struct{
  int Fix;
} Str[] = { 100,200 },
  *Ptr  = Str;

main()
{
  printf("%d", ++ Ptr ++ -> Fix );
  printf("%d", --- Ptr -> Fix   );
}
```

Wydruk 2. Program niepoprawny

Uwaga

W niektórych implementacjach, np. Aztec i MicroSoft, zezwala się, aby przed operatorem „—" występowało wyrażenie, które nie jest pierwotne. W takim przypadku program zamieszczony na wydruku 2 zawiera błąd jedynie w drugiej instrukcji printf.

Ogłoszenia • Ogłoszenia • Ogłoszenia • Ogłoszenia

Programy, instrukcje i udoskonalenia techniczne dla komputerów ATARI, COMMODORE, IBM oferuje Agencja Komputerowa. 41-200 Sosnowiec, p-157, tel. 63-29-35.

EO/423/87

Ogłoszenia • Ogłoszenia • Ogłoszenia • Ogłoszenia

CeBIT '87

dokończenie ze str. 46

kserograficznych. Po przekazaniu Comparexowi stoiska w hali nr 1, produkty te BASF eksponował w nowym obszernym stoisku w hali nr 4.

CeBIT jako chronologicznie pierwsza w roku, a jednocześnie największa na świecie cykliczna ekspozycja informatyczna, jest tradycyjnie okazją do prezentowania przez wszystkie firmy nowości, zwłaszcza tych o charakterze przełomowym. Nowości i tym razem było bardzo dużo (ok. 45% całości ekspozycji), przy czym dotyczyły one głównie zwiększenia rozdzielczości ekranów, poprawy technicznych i ekonomicznych parametrów drukarek (zwłaszcza laserowych), wzrostu pojemności i szybkości pamięci operacyjnych i masowych oraz rozszerzenia zakresu funkcji i zwiększeniu wygody oprogramowania użytkowego. Nowości te nie stanowiły jednak domeny pojedynczych firm, lecz powtarzały się z niewielkimi odchyleniami parametrów u znacznej liczby wystawców. Świadczy to zarówno o bardzo ostrej walce konkurencyjnej, jak i trudnościach zapobiegania przeciekom postępu technologicznego. Nowości o prawdziwie przełomowym charakterze właściwie nie było, bo nawet IBM, mimo wystąpienia z wyjątkowo rozległą, wielostanowiskową ekspozycją aż w 5 halach, dopiero w miesiąc po CeBIT zaprezentował swój podstawowy szlagier 1987 r.: dawno oczekiwany system PS/2.

Na CeBIT '87 powtórzyła się wyraźna już w roku poprzednim tendencja do akcentowania w przytłaczającej większości eksponatów ich pełnej kompatybilności z produktami IBM (nie tylko zresztą w obszarze sprzętu mikrokomputerowego), a także powszechnej integracji komputerów z techniką telekomunikacyjną. Zwłaszcza ta ostatnia tendencja w mniej zaawansowanym obserwatorze ekspozycji mogła ukształtować fałszywe (u nas niestety powszechne) wyobrażenie o zaniku rynku średnich i dużych systemów komputerowych, a nawet minikomputerowych, oraz ich zastępowania przez szybko rozwijający się sprzęt mikrokomputerowy, zwłaszcza w klasie konstrukcji 32-bitowych. W stoiskach wszystkich producentów, nie wyłączając dostawców największych maszyn, eksponowano bowiem tylko mikrokomputery, które niewątpliwie są najlepszym z dostępnych środków pozyskiwania nowych klientów.

Rozwój telekomunikacji doprowadził do całkowitej rezygnacji przez wystawców z kosztownej i bardzo trudnej w warunkach targowych ekspozycji dużych maszyn. Stąd znaczna część wystawionych mikrokomputerów jest po prostu inteligentnymi urządzeniami końcowymi, przyłączonymi do dużych systemów zainstalowanych poza terenem targowym. Rozwiązania takie umożliwiają prezentowanie szczególnie atrakcyjnych przykładów zastosowań.

Tak jak poprzednie imprezy, CeBIT '87 był ukierunkowany na olbrzymi rynek małych przedsiębiorstw. Mimo szybko postępującej obniżki cen oraz wzrostu efektywności sprzętu i oprogramowania mikrokomputerów, informatyka dociera tu bardzo powoli. Użytkowników tej kategorii z jednej strony nie stać na angażowanie specjalistów, z drugiej zaś nie należą jeszcze do pokolenia przygotowanego do akceptacji techniki komputerowej. Dlatego tak istotnym punktem programu Targów był ponownie Computer Camp (Obóz komputerowy), adresowany do szerokich kręgów młodzieży RFN. Jest to ciąg atrakcyjnych imprez poznawczych i promocyjnych, zachęcających do zainteresowania się informatyką i wyboru kariery zawodowej w tym właśnie kierunku.

Istotnym punktem ekspozycji ze względu na wagę zagadnienia były w hali nr 17 pokazy działania sieci lokalnej Multinet, akceptującej sprzęt i oprogramowanie kilkunastu producentów. Oznacza to, że prowadzona uporczywie normalizacja rozwiązań czyni już widoczne postępy.

CeBIT '88 odbędzie się w terminie od 16 do 23 marca, a więc w kolizji z Wiosennymi Targami Lipskimi (13—19 marca). Z decyzji tej wynika, że organizatorzy imprezy hanowerskiej nie mają już obaw o wielkość frekwencji.

WŁADYSŁAW KLEPACZ

Język C — propozycja polskiej terminologii (1)

Utrzymujące się zainteresowanie językiem C uzasadnia podjęcie dyskusji na temat jego polskiej terminologii. Ponieważ podanie suchych definicji poszczególnych pojęć, w oderwaniu od użycia ich w kontekście, z pewnością zawęziłoby dyskusję, wiele z proponowanych tu terminów użyto w zdaniach odnoszących się do odpowiednio dobranych konstrukcji językowych.

agregat

Agregatami są tablice, struktury i unie.

argument

fun (len, -vol, size + base)

Argumentami wywołania funkcji są: len, -vol i size + base. Argumentami operatora „—" (minus) jest vol, a argumentami operatora „+" (plus) są size i base.

ciało funkcji

```
fun (par)
  char par;
{ return !!par; }
```

Ciałem funkcji jest instrukcja grupująca następująca po nagłówku funkcji i deklaracjach jej parametrów.

dana

Dana jest to obiekt podlegający przetwarzaniu. Dane dzielą się na arytmetyczne (stałopozycyjne i zmiennopozycyjne), znakowe i wskazujące. Dane stałopozycyjne i znakowe są obejmowane wspólną nazwą danych całkowitych. W programach dane są reprezentowane przez wyrażenia, których szczególnym przypadkiem są literały: stałopozycyjne (24), zmiennopozycyjne (24.0), znakowe ('j') i napisowe ("jb"). W języku C nie zdefiniowano literału wskazującego. Jego rolę pełni literał całkowity 0.

dana wskazująca

Dana wskazującą jest dana wskazująca inną daną.

```
int Fix = 13,
    *Ptr = &Fix,
    **Ref = &Ptr;
main () {
  printf ("%d", **Ref);
}
```

Zmiennej Ptr przypisano daną wskazującą zmienną Fix, a zmiennej Ref przypisano daną wskazującą zmienną Ptr.

deklaracja

Deklaracje dzielą się na definiujące i odwoławcze. Deklaracje definiujące są nazywane definicjami. Większość deklaracji ma charakter jawny, ale w języku C istnieją także deklaracje domniemane i kontekstowe.

```
fun par
{ return sub (par); }
```

Definicja funkcji fun implikuje domniemaną deklarację parametru

```
int par;
```

i kontekstową deklarację funkcji sub

```
external int sub ();
```

deklarator

Deklaratory dzielą się na deklaratory funkcji i danych. Deklaratory służą do deklarowania atrybutów związanych z identyfikatorami. Deklarator funkcji występujący w nagłówku funkcji deklaruje ponadto identyfikatory pełniące role parametrów funkcji

```
int fun(max,min)
  float max,min;
{ return max == min; }
```

Pierwszy wiersz definicji funkcji fun zawiera oznaczenie typu i deklarator funkcji, natomiast drugi zawiera deklaracje typu parametrów.

epilog

Epilog jest to zestaw czynności wykonywanych tuż przed zakończeniem wykonywania bloku wyznaczonego przez instrukcję grupującą, albo tuż przed zakończeniem wykonywania programu. Stosownie do sytuacji mówi się o epilogu instrukcji grupującej albo o epilogu programu.

```
#include "stdio.h"
main () {
  FILE *fp,*fopen ();
  if((fp = fopen("empty", "w")) == NULL)
    exit(1);
}
```

Wykonanie epilogu instrukcji grupującej powoduje wyeliminowanie zmiennej identyfikowanej przez fp. Wykonanie epilogu programu powoduje zamknięcie pliku (o ile został otwarty).

identyfikator

Identyfikatorem jest nazwa symboliczna składającą się z liter i cyfr, rozpoczynająca się od litery. Pojęcie litery obejmuje także znak podkreślenia.

```
main () {
}
```

Nazwa symboliczna main jest identyfikatorem funkcji.

inicjalizacja

Inicjalizacja jest to czynność polegająca na przypisaniu zmiennej, danej początkowej. Inicjalizacja może być jawna albo niejawna.

```
fun (par) {
  auto exp = par;
  static imp;
  return exp + imp;
}
```

Zmienna exp podlega inicjacji jawnej, a zmienna imp inicjacji niejawnej.

inicjator

```
char arr [] = { "Jan", "Ewa" };
```

Napis występujący między znakami „=" (równość) i „;" (średnik) jest inicjatorem.

instrukcja

Instrukcjami języka C są: instrukcja pusta (;), przejścia (goto), grupująca ({}), warunkowa (if), cyklu (for, while, do), kontynuowania (continue), zaniechania (break), decyzyjna (case) i powrotu (return). Instrukcją jest także dowolne wyrażenie zakończone średnikiem.

jednostka leksykalna

Jednostkami leksykalnymi są: identyfikator (Jan), słowo kluczowe (if), literał ('E'), operator (++) oraz separator (;). Do reparowania jednostek leksykalnych służą komentarze i odstępy.

klasa danych

Dane w języku C dzielą się na automatyczne (klasa auto), rejestrowe (klasa register), statyczne (klasa static) i zewnętrzne (klasa extern).

kolizja identyfikatorów

```
fun (par)
  float par;
{ char par = 'j';
  return par;
}
```

Kolizja identyfikatora parametru i identyfikatora zmiennej lokalnej ciała funkcji powoduje ograniczenie zasięgu deklaracji parametru do nagłówka funkcji.

komentarz

Napis

```
/* To be or not to be, that is the question */
```

jest komentarzem.

konwersja

Konwersja jest przekształcenie danej pewnego typu na daną tego samego albo innego typu. Konwersje mogą być

jawne — zadawane za pomocą operatora konwersji, albo niejawne. Przykładem konwersji niejawnych są typowe konwersje arytmetyczne.

```
long int fun (par)
  float par;
{ int var = par;
  return (float) var;
}
```

Podczas przypisywania danej początkowej zmiennej var dokonywana jest typowa konwersja arytmetyczna. Podczas wykonywania instrukcji return wykonywana jest jawna konwersja na daną typu (float) oraz niejawna konwersja na daną typu (long int).

literał

Literałem jest taki napis reprezentujący stałą, z którego postaci (w ramach ustalonej interpretacji) wynikają wszystkie atrybuty tej stałej, wraz z jej wartością. Napis:

"Jan"

jest literałem napisowym. Wartością stałej reprezentowanej przez ten literał jest wskazanie ciągu 4 znaków, z których ostatni jest znakiem o kodzie 0.

l-wyrażenie

Wyrażenie, które może zostać użyte w znaczeniu lewego argumentu operatora przypisania jest l-wyrażeniem.

```
——(* ( ptr ) ref ? ptr : ref)) ;
```

Argumentem operatora „——" (podwójny minus) jest l-wyrażenie.

Argumentem operatora „*" (gwiazdka) nie jest l-wyrażenie.

makrodefinicja

```
# define macro(min,max) ((min) == (max))
```

Identyfikator macro jest nazwą makrodefinicji o parametrach min i max.

makrosubstytucja

Makrosubstytucja polega na zastąpieniu makrowywołania ciągiem jednostek leksykalnych zawartych w makrodefinicji.

makrowywołanie

```
#define head(par) par
#define tail {}
head(main () tail
```

W następstwie zastosowania makrosubstytucji, makrowywołanie head(main ()) zostanie zastąpione napisem main (), a makrowywołanie tail zostanie zastąpione napisem {}.

nadanie wartości

Określenie „nadanie wartości" jest skrótem od „przypisanie (zmiennej programu) danej o ustalonej wartości".

```
int var = 5;
```

Opracowanie przytoczonej deklaracji powoduje przypisanie zmiennej var danej o wartości literału 5, w skrócie: nadanie zmiennej var wartości początkowej 5.

nagłówek funkcji

```
fun (par)
  char par;
{ return ! ! par; }
```

Nagłówek funkcji fun składa się z deklaratora funkcji zawierającego listę parametrów funkcji.

napis

Napisem jest dowolny ciąg znaków, w szczególności literał i identyfikator, ale także — dowolne ich części.

numer pliku

```
main ()
{ close (open("kaja",1)) ; }
```

Argumentem funkcji close jest wyrażenie reprezentujące numer pliku.

odstęp

Odstępem jest dowolny ciąg spacji, znaków tabulacji i znaków nowego wiersza. Odstępy i komentarze służą do separowania jednostek leksykalnych.

operacja

Operacją jest czynność wyrażona za pomocą operatora. Operacje dzielą się na jednoargumentowe, dwuargumentowe i trójargumentowe.

JAN BIELECKI

WARUNKI PRENUMERATY NA 1988 R.

Prenumeratorzy zbiorowi — jednostki gospodarki uspołecznionej, instytucje i organizacje społeczne zamawiają prenumeratę dokonując wpłaty wyłącznie na blankiecie „wpłata-zamówienie" (jest to „polecenie przelewu" rozszerzone dla potrzeb Wydawnictwa o część dotyczącą zamówienia). Blankiety te będą dostarczane dotychczasowym prenumeratorom przez Zakład Kolportażu. Nowi prenumeratorzy otrzymają je po zgłoszeniu zapotrzebowania (pisemnie lub telefonicznie) w Zakładzie Kolportażu.

Prenumeratorzy indywidualni — osoby fizyczne zamawiają prenumeratę dokonując wpłaty w UPT lub NBP na blankiecie NBP. Na odwrocie wszystkich odcinków blankietu należy wpisać tytuł czasopisma, okres prenumeraty, liczbę zamawianych egzemplarzy oraz wartość wpłaty. Wpłacać należy na konto: NBP III Oddział Warszawa 1036-7490-139-11.

Prenumerata ulgowa — przysługuje wyłącznie osobom fizycznym — członkom SNT, studentom i uczniom szkół zawodowych. Warunkiem prenumeraty ulgowej jest poświadczenie blankietu wpłaty (przed jej dokonaniem) na wszystkich odcinkach pieczęcią Koła SNT, wyższej uczelni lub szkoły. Sposób zamawiania prenumeraty ulgowej jest taki sam jak prenumeraty indywidualnej. W prenumeracie ulgowej można zamówić tylko po 1 egzemplarzu każdego czasopisma.

Uwaga! Miesięcznik „Aura" może być zamawiany w prenumeracie ulgowej również przez uczniów szkół ogólnokształcących.

Prenumerata ze zleceniem wysyłki za granicę — zamawia się tak jak prenumeratę indywidualną. Dodatkowo należy podać na blankiecie wpłaty nazwisko i dokładny adres odbiorcy. Cena prenumeraty ze zleceniem wysyłki za granicę jest dwukrotnie wyższa.

Wpłaty na prenumeratę przyjmowane są w terminach:

- do 10 listopada na każdy kwartał, I i II półrocze oraz cały rok następny,
- do 28 lutego na II, III i IV kwartał oraz II półrocze,
- do 31 maja na III i IV kwartał oraz II półrocze,
- do 31 sierpnia na IV kwartał.

Zmiany w prenumeracie można zgłaszać pisemnie tylko w wyżej wymienionych terminach.

Informacji o prenumeracie udziela — Zakład Kolportażu Wydawnictwa NOT SIGMA (ul. Bartycka 20, 00-716 Warszawa), skr. poczt. 1004, 00-950 Warszawa, tel. 40-00-21 wew. 248, 249, 293, 297, 299 lub 40-30-86 i 40-35-89.

Egzemplarze archiwalne czasopism — można nabyć za gotówkę w Klubie Prasy Technicznej, Warszawa ul. Mazowiecka 12 (tel. 27-43-65), lub zamówić pisemnie. Zamówienia na egzemplarze archiwalne czasopism przyjmuje: Zakład Kolportażu, Dział Handlowy, 00-950 Warszawa, skr. pocz.t 1004 (tel. 40-37-31), na rachunek dla instytucji lub za zaliczeniem pocztowym dla osób fizycznych.

Cena prenumeraty wg cennika na 1988 rok					
kwartalna		półroczna		roczna	
normalna	ulgowa	normalna	ulgowa	normalna	ulgowa
600,—	150,—	1200,—	300,—	2400,—	600,—

Cena egzemplarza 200 zł (50 zł — cena ulgowa).

Konferencje

INFOPRO '87

Konferencje INFOPRO poświęcone zastosowaniom informatyki w projektowaniu budownictwa mają już dziesięcioletnią tradycję. Pierwsza odbyła się w 1977 r., następne w cyklu dwuletnim.

INFOPRO'87 było już szóstą konferencją z tego cyklu, którego inicjatorem i organizatorem jest Centralny Ośrodek Badawczo-Projektowy Budownictwa Przemysłowego BISTYP w Warszawie. Współorganizatorem konferencji tegorocznej, która odbyła się w dniach 6—8 maja br. w Kudowie, był Ośrodek Współpracy w Projektowaniu Budownictwa oraz Komitet Projektowania Zarządu Głównego PZITB. Konferencja zgromadziła 340 uczestników, głównie z biur projektów i wyższych uczelni. Podstawowe sesje obejmowały referaty i dyskusje na temat zastosowań informatyki w projektowaniu architektury, konstrukcji budowlanych, instalacji sanitarnych i elektrycznych, ochrony środowiska, w kosztorysowaniu oraz zarządzaniu biurami projektów. Odbyło się również otwarte zebranie Klubu Użytkowników Komputerów w Pracach Projektowych i Badawczych, a także przeprowadzono dyskusję na temat przyszłego modelu zastosowań informatyki w projektowaniu. Obradom towarzyszył pokaz sprzętu i oprogramowania, na którym 11 przedsiębiorstw prezentowało swoją ofertę adresowaną głównie do biur projektów.

Konferencje INFOPRO stanowią znaczące w skali kraju forum wymiany informacji o stosowanych w projektowaniu nowych programach i systemach oraz o prowadzonych pracach badawczych i metodologicznych, a także przyczyniają się do kształtowania poglądów na kierunki dalszego rozwoju. Stanowi to istotny element integracji środowiska i stymuluje tempo i prawidłowe kierunki rozwoju zastosowań informatyki w projektowaniu.

Przez wiele lat głównym problemem biur projektów był brak sprzętu komputerowego. Ścierały się poglądy, czy lepiej mieć własny komputer, czy też korzystać z usług ośrodków obliczeniowych. Stopniowo wykształcił się pogląd, że racjonalne stosowanie informatyki w projektowaniu wymaga stałego, bezpośredniego i natychmiastowego dostępu do komputera, a rozwiązaniem optymalnym dla biur projektów są minikomputery. Pojawiła się MERA 400 wspólnie wdrażana i oprogramowywana w tym właśnie kierunku. Konferencje INFOPRO odegrały istotną rolę w formułowaniu postulatów kierowanych do producenta MERY i przyczyniły się do szybkiego wdrożenia tych minikomputerów.

W ostatnich latach, dzięki możliwościom zakupu mikrokomputerów powstała w zakresie sprzętu całkowicie nowa sytuacja. Biura projektów wyposażają się szybko w liczne egzemplarze sprzętu typu IBM/PC lub Amstrad (niekiedy także Spectrum). W konsekwencji tego faktu szybko wzrasta liczba aktywnych użytkowników komputerów oraz zmienia się model organizacyjny ich stosowania. W ubiegłych latach utrwaliło się korzystanie z metod informatycznych za pośrednictwem specjalistów tworzących w biurze zespół, pracownię lub ośrodek obliczeniowy. Projektanci na ogół pozostawali biernymi użytkownikami. Nie stykali się oni bezpośrednio z komputerem, przekazując tylko dane do obliczeń i odbierając wydruki wyników, które stanowiły materiał do projektowania tradycyjnego. Taki model organizacyjny był logiczną konsekwencją rodzaju stosowanego sprzętu, trudnego w obsłudze dla niespecjalistów oraz kosztownego, a więc niechętnie im udostępnianego z obawy przed uszkodzeniami lub nieracjonalnym wykorzystywaniem.

Wejście w epokę mikrokomputerów otwiera nowe możliwości i nowe potrzeby. Zamiast jednego komputera w ośrodku informatycznym instaluje się wiele egzemplarzy bezpośrednio na stanowiskach pracy. Decentralizacja sprzętu zmienia stosunek do informatyki projektantów, którzy samodzielnie korzystając z mikrokomputera muszą opanować niezbędny zakres umiejętności jego obsługi. Wydaje się jednak, że w przeciętnym biurze projektów nadal niezbędne będzie istnienie ośrodka czy zespołu informatycznego, przygotowującego oprogramowanie oraz pomagającego we wdrażaniu nowych rozwiązań.

Jak oceniono w czasie konferencji — w ostatnich latach przebiegał systematyczny, lecz stosunkowo powolny w porównaniu do rewolucyjnego postępu w zakresie sprzętu, rozwój oprogramowania. Nastąpił odczuwalny wzrost liczby dostępnych programów oraz doskonalenie ich cech użytkowych. Rozszerzał się powoli zakres zastosowań grafiki komputerowej. Nadal nieliczne są jednak programy zawierające procedury optymalizacyjne, a także programy symulacyjne. Nie nastąpił też szerszy rozwój zastosowań w dziedzinie zarządzania i obsługi procesu projektowania. Nienadążanie oprogramowania wymaga intensywnego tworzenia programów uwzględniających możliwości współczesnego sprzętu. Potrzebne są zwłaszcza programy z procedurami optymalizacyjnymi, symulujące działanie urządzeń i instalacji oraz oparte na nowych algorytmach z zakresu problematyki projektowania funkcji i przestrzeni.

Niemal od podstaw trzeba tworzyć sprawne systemy obsługi biura projektów w zakresie planowania, organizacji i kontroli produkcji, rozliczeń i księgowości, obsługi archiwów i całego zespołu informacji technicznej, a także korespondencji. Specyfika działania przedsiębiorstw projektowych sprawia, że na ogół nie udaje się w nich zastosować systemów wykorzystywanych w przedsiębiorstwach innych branż.

Rozwój mikrokomputerów sprawił, że dość wyraźnie rysuje się już przyszłościowy model skomputeryzowanego biura projektów, któremu poświęcono specjalną sesję dyskusyjną konferencji. Przyszłość, to sieć z mikrokomputerem lub końcówką na każdym stanowisku jako podstawowym narzędziem pracy. Wszystkie operacje procesu projektowania będą realizowane opierając się na biurowych, regionalnych lub krajowych bankach danych, zawierających większość informacji, z których korzysta w pracy projektant. Realizacja modelu przyszłościowego sprowadza się głównie do oprogramowania oraz banków danych.

W referatach i dyskusjach wyrażano opinię, że uzyskanie wyższej jakości oprogramowania będzie możliwe dopiero po opracowaniu zasad przepływu informacji oraz zawartości banków danych. Niezbędne jest podjęcie podstawowych prac badawczych i metodologicznych dotyczących problematyki procesu projektowania oraz powiązań projektowania z procesem inwestycyjnym.

Kompleksowa komputeryzacja rysunkowej części projektów (zwłaszcza konstrukcji) będzie prawdopodobnie wymagać wprowadzenia istotnych zmian w sposobach graficznej prezentacji konstrukcji, a następnie systematyzacji zapisu graficznego. Dopiero na tej podstawie możliwe będzie opracowanie systemów umożliwiających komputerowe sporządzanie wszelkich rysunków, w tym również dla rozwiązań indywidualnych i nietypowych.

Konferencja wykazała, że nadal niewystarczająca jest informacja o istniejących programach, a także o pracach nad nowymi programami. Tak jak przed 10 laty, nadal nie rozwiązane są problemy praw autorskich do programów oraz odpowiedzialności za poprawność programów i obliczeń komputerowych.

Postulowano wprowadzenie atestacji programów do obliczeń inżynierskich, co wpłynęłoby na istotny wzrost zaufania użytkowników do metod komputerowych.

Na przełomie lat 1986—1987 uruchomiono liczne prace badawczo-rozwojowe z dziedziny informatyki, w tym także dotyczące metod komputerowych w projektowaniu. Można mieć nadzieję, że w ramach tych prac wiele z omawianych na konferencji problemów znajdzie prawidłowe rozwiązanie.

Konferencja odbywała się w atmosferze zwiększonego zainteresowania całego społeczeństwa. Stąd o udział w posiedzeniach i pokazach ubiegali się również mieszkańcy i kuracjusze Kudowy, w tym także młodzież szkolna, dla której zorganizowano specjalne pokazy.

Konferencja jeszcze raz potwierdziła swoją przydatność dla środowiska związane zwiększonego zainteresowania informatyką całego społeczeństwa. Stąd o udział w posiedzeniach i pokazach ubiegali się również mieszkańcy i kuracjusze Kudowy, w tym także młodzież szkolna, dla której zorganizowano specjalne pokazy.

Dr inż. MACIEJ ROBAKIEWICZ
COBPBP „BISTYP"

Konferencje

INFOSYSTEM '87

Ekspozycja komputerowa na Międzynarodowych Targach Poznańskich co najmniej od 1983 roku ciągle rosła. Komputery zajmowały początkowo fragment pawilonów Metronexu i krajów ościennych, potem także pawilon firm polonijnych, a następnie samodzielną, sporą halę. Wreszcie nadszedł czas na wyodrębnienie wystawy komputerowej z ogólnych Targów, tak jak to poprzednio stało się, na przykład ze sprzętem medycznym i poligrafią. W tym roku nastąpiło to po raz pierwszy, dzięki inicjatywie „Przeglądu Technicznego" (z okazji 120-lecia gazety) oraz Elwro. Ze względu na współorganizatora wystawa, nazwana INFOSYSTEM'87, mieściła się w Hali Ludowej we Wrocławiu, na powierzchni 4 tys. m². Wystawiało 80 firm krajowych oraz 22 firmy zagraniczne (w rzeczywistości pewnie więcej, gdyż czasami w jednym stoisku pod jedną wspólną nazwą występowało kilku wystawców). Zapowiedziano, że od przyszłego roku wystawa wejdzie na stałe do kalendarza targowego i odbędzie się na terenach targowych w Poznaniu w dniach 25—29 kwietnia.

Wymowa wystawy była zupełnie jednoznaczna. Oceniając na oko, 90% sprzętu stanowiły komputery osobiste, z których 90% to kopie IBM PC XT AT. Było troche komputerów domowych („grybasów"), ale stanowiły one margines. Trafna wydaje się żartobliwa opinia, że wystawę powinien współotwierać przedstawiciel ministerstwa przemysłu i handlu Tajwanu, gdyż był to najwiekszy pokaz tajwańskich komputerów w Europie. Tylko nieliczne firmy pokazywały sprzęt innej proweniencji, choć oczywiście trudno to ocenić z całkowitą pewnością bez zaglądania do środka.

Imponujące wrażenie robił ciąg stoisk zajmowanych przez firmy, które na nasz użytek nazwę koncernem Kajkowskich. Firma, która 5 lat temu zaczynała od sprzedaży 8-bitowych komputerów Lidia, obecnie rozwinęła się w całą sieć kooperujących przedsiębiorstw, przygotowujących i sprzedających sprzęt (typowy, tzn. IBM) i oprogramowanie — własne i przerobione, mniej lub bardziej spolszczone.

Mieszane uczucia budziło stoisko spółki Mikrokomputery, w którym prezentowano komputer Mazovia 1016. Był to praktycznie pierwszy publiczny pokaz tego sprzętu — dotychczas można go było tylko podziwiać z zewnątrz i oceniać wygląd obudowy. Mazovia stanowi odpowiednik IBM PC, lecz korzysta z procesora 8086. Szesnastobitowa szyna danych daje wiekszą wydajność przy nie zmienionym cyklu zegara, ale wymaga dodatkowych układów dopasowania umożliwiających dołączenie pakietów zgodnych ze standardem IBM. Zapewne projektanci liczyli na dostawy procesorów 8086 z ZSRR, ale widać nic z tego nie wyszło, skoro na wystawianych pakietach wszystkie procesory i inne układy o dużym stopniu scalenia pochodziły z Zachodu (którego zasadnicza cześć leży na Dalekim Wschodzie). Chlubnym wyjątkiem był port równoległy 8255 produkcji CEMI i kilka układów pamięci dynamicznej 64 KB w wykonaniu militarnym (z obudową ceramiczną) z ZSRR. Te ostatnie umieszczono tylko na jednym pakiecie i chyba normalnie nie będą używane. Wprawdzie Elorg wystawił w gablotce układy, na których było napisane 256, ale — nawet jeśli nie były to atrapy — zapewne radziecki rynek sam wchłonie każda wyprodukowana ilość. O rozwiązaniach technicznych Mazovii będziemy pisać osobno, obecne ceny wynosza 2,5 mln zł za komputer z pamięcia 256 KB RAM, 2 stacie dyskietek i jednobarwny monitor (niestety — bardzo kiepskiej jakości; nie chciałbym siedzieć przy nim przez 6 godzin dziennie!). Dodatkowe 256 KB RAM kosztuje milion złotych! Stacje dyskietek (a w perspektywie także dyski sztywne i streamery) pochodza z importu. Drukarka D-100E potrafi udawać drukarkę Epson FX80, choć niestety zachowuje osobliwie niefortunny sposób przesuwania papieru oryginalnej drukarki D-100. Oprogramowanie dla Mazovii obejmuje spolszczone wersje programów Wordstar (nazwanego Redaktor), dBase, Lotus 1-2-3, pfs i innych. Zapewne także inne programy dadzą się przenieść na ten sprzęt.

Polski wielki przemysł prezentował także duże komputery Jednolitego Systemu z Elwro i przebój sezonu — Elwro 800 Junior, przeznaczony dla szkół. Katowicki Meraster przedstawił bardzo ładne i poręczne terminale do Mery 660, udające VT 52, po 280 tys. zł, a Meramat — streamery na kasety 1/4 cala, po 350 tys. zł, dostępne podobno od stycznia przyszłego roku.

Także centrale handlowe innych krajów RWPG prezentowały mniej lub bardziej udane kopie IBM PC. Tylko Czechosłowacja (KOVO) prezentowała ciągle rozwijana wersję SM, m.in. SM 52/11 (klasy PDP-11/70), w zestawieniu z deską kreślarską i oprogramowaniem typu CAD, np. do celów kartograficznych. Niestety nie pokazywano odpowiednika Vaxa, choć już w tym roku pierwszy model ma się pojawić w warszawskim Zowarze. Szkoda tylko, że wszystkie te niezłe minikomputery korzystają ciągle z fatalnych bułgarskich stacji dyskowych, co podaje w wątpliwość ich niezawodność.

Wróćmy do standardu IBM PC. Kto żyw starał się prezentować rozwiązania rozproszone, wielostanowiskowe, poczynając od terminali dołączanych do komputera osobistego klasy XT lub AT. Po uruchomieniu protezy, np MultiLink (firma DataCo) lub prawdziwego wielodostępnego systemu operacyjnego (Unix, Xenix — np. ComTech), istotnie każdy użytkownik ma dostęp do zasobów komputera, jednak: nie każdy program będzie pracował w tym trybie; do podziału pozostanie niewiele pamięci; programy nie będą wzajemnie przed sobą chronione; małe, tanie dyski Winchester z silnikami krokowymi nie wytrzymają zbyt intensywnej pracy. Można więc traktować taki system jako zamiennik dla Seechecka (Mera 9150), ale pod warunkiem samodzielnego napisania programu obsługi terminali, bo np. dwie kopie dBase nie zmieszczą się w pamięci. Niektóre firmy oferują jako terminale komputery 8-bitowe (np. ComTech — komputery Spectra-Video, udające terminale VT 52, Biurotechnika — komputery Meritum).

Większe możliwości daje lokalna sieć łącząca samodzielne komputery. Do małych konfiguracji oferowano względnie powolne sieci korzystające ze zwykłego, dwużyłowego kabla: Transnet (firma Promax spolszczyła oprogramowanie wspomagające) i D-Link (np. DataCo). Następny poziom — to oryginalna sieć PC Net (Polmer, 1,9 mln zł na jeden komputer). Wreszcie ciężka artyleria — Ethernet na kablu koncentrycznym, na ogół w połączeniu z większymi komputerami (Globo — wraz z profesjonalnym mikrokomputerem na procesorze 80386 i solidnym dyskiem w sumie za 22 mln zł).

Okazuje się, że mimo faktycznego czy mitycznego embarga da się sprowadzić do Polski komputery 32-bitowe. Firmy Komplex i Globo wystawiły komputery z procesorem 80386. Reflex — pakiet wstawiany do IBM PC, wyposażony w procesor 68020 i kilka magabajtów pamięci, Arton — takiż pakiet, a w perspektywie może inne 32-bitowe rozszerzenia. Optymiści twierdzili, że wstawienie takiego pakietu pozwala na pracę wielodostępna, np. z jakaś wersja Unixa. Realiści przedstawili to tylko jako ułatwienie i przyspieszenie pracy jednego użytkownika.

Spośród znanych firm zachodnich najokazalej wystąpiły ICL i Hewlett-Packard. Firma ICL tradycyjnie już wystawia wysokiej klasy sprzęt mini- i mikrokomputerowy dla biur, banków itp. Tym razem były to trzy modele. Mikrokomputer DRS 300 ma procesor 80286, co najmniej 1 MB RAM, od 20 do 320 MB pamięci dyskowej i pracuje pod kontrolą systemu PC-DOS 4.1 lub Unix V.2. W tym drugim wypadku może obsługiwać do 16 terminali. Cena — od 5 tys. funtów. Mikrokomputer CLAN ma procesor 68010, do 8 MB pamieci RAM i 450 MB na dyskach. Pod nadzorem Unixa 5.0 obsługuje do 16 terminali (cena — 25 tys. funtów). Najmniejszy model, komputerek osobisty mający stanowić wyposażenie biurowe i połączony z telefonem, korzysta z procesora 68008 (znanego do tej pory bodaj wyłącznie z niesławnego Sinclair QL) i kosztuje 800 funtów. Do tego dochodzą sieci o różnej technologii i

przepustowości od 0,5 do 10 milionów bitów na sekundę.

Hewlett-Packard był reprezentowany przez firmę ZOTPAN, która zresztą wkrótce otworzyła w Warszawie nowy ośrodek obliczeniowy. Przedstawiono profesjonalne komputery z serii 3000, przeznaczone do zarządzania produkcją i dla dużych przedsiębiorstw handlowych. Najmniejszy model, za 45 tys. dolarów ma 2 MB RAM, 81 MB pamięci dyskowej, drukarkę i 7 terminali. Była też drukarka laserowa, dająca oczywiście jakość druku niedoścignioną na innych urządzeniach, ale wymagającą w eksploatacji wsadu dewizowego (ok. 4 centów na stronę).

Spośród urządzeń zewnętrznych, oczywiście najszerzej prezentowano drukarki — od standardowych i tanich drukarek mozaikowych 9-igłowych, przez znacznie lepsze 24-igłowe (np. doskonała drukarka Star NB 24 wystawiana przez firmę ABC Data), a kończąc na kolorowej termicznej drukarce mozaikowej Mitsubishi (wystawiała firma Quest) o rozdzielczości 100 punktów na mm² i szybkości około 1 strony na minutę. Jej jakość jest dostosowana do karty grafiki EGA — znany rysunek mandryla po wydrukowaniu wygląda jak żywy! Oczywiście cena odpowiada efektowi — 5,5 tys. funtów.

Firma Rotring, znany producent rapidografów i desek kreślarskich (a właściwie jej przedstawiciel — Promax), prezentowała dwa modele mikroploterów, mocowanych na desce kreślarskiej obok normalnych linijek. Wyposażone w małą klawiaturę i uchwyt rapidografu, plotery te potrafią pisać pismem technicznym teksty o różnej wielkości i różnym nachyleniu, kreślić proste figury (odcinki, łuki) różnymi rodzajami linii, a także rysować symbole z biblioteki, np. sanitarne i elektryczne. Po połączeniu łączem szeregowym z komputerem ich możliwości jeszcze rosną.

Większość prezentowanych komputerów była wyposażona w standardowe monitory jednobarwne lub kolorowe klasy EGA. Firma Conrac prezentowała monitor (i sterownik) dla potrzeb projektowania, o rozdzielczości 1280 × 1024 punkty.

Firma Prosystem pokazała urządzenie doskonale nadające się do wszelkich sal wykładowych — wyświetlacz na ciekłych kryształach przystosowany do rzutników folii (jak np. polski Lech). Zamiast folii kładzie się na płytę rzutnika wyświetlacz, który pokazuje dokładnie to, co widać na ekranie komputera.

Każda firma prezentowała jakieś oprogramowanie. Na ogół były to wersje standardowych programów (Wordstar, dBase itp.) w różnym stopniu spolszczone. Czasami tylko komunikaty były po polsku, czasami także polecenia lub ich część. Z reguły nie korzystano z polskich liter (lub liter takich po prostu nie było), lub korzystano niekonsekwentnie (np. koncern Kajkowskich tylko w niektórych programach korzysta z polskich znaków). Były też oczywiście programy wprowadzania i przetwarzania danych magazynowych, finansowo-księgowych itp., stanowiące w gruncie rzeczy scenariusze dla programu dBase.

Było także trochę polskiej dokumentacji. Najlepszą chyba wystawiła firma Org-service, np. podręczniki Turbo Pascala (za 5,5 tys. zł) i DOS-a (za 2,2 tys. zł). Dla porównania podręcznik użytkownika Turbo Pascala firmy Intersoft jest oferowany za 67 tys. zł!

JAROSŁAW DEMINET

Głosowy dostęp do baz danych

Firmy amerykańskie Artificial Intelligence Corporation (AIC) oraz Kurzweil AI Inc. zawarły porozumienie handlowe, którego celem jest praktyczne zintegrowanie technik dostępu opartego na języku naturalnym z technikami dostępu głosowego. Te dwie uzupełniające się techniki umożliwią w przyszłości budowę bardziej naturalnych środków komunikacji między użytkownikiem a komputerem.

Dostęp w języku naturalnym

Systemy wyszukiwania informacji oparte na dużych komputerach stały się jedną z pierwszych dziedzin wykorzystania technik dostępu w języku naturalnym. W tradycyjnych systemach wyszukiwania użytkownik musi formułować pytania według ściśle określonych reguł języka zapytań bądź też konstruować odpowiednie programy. Techniki menu upraszczają nieco to zadanie, lecz jednocześnie zmniejszają elastyczność zastosowań, zmuszając użytkownika do powtarzania — ustalonego z góry — ciągu operacji. Poszukiwano więc takich metod dostępu, które zwiększyłyby elastyczność i prostotę zadawania pytań. Właściwym rozwiązaniem tego zadania okazało się wykorzystanie technik komunikacji w języku naturalnym.

Programy, które rozumieją polecenia formułowane w języku naturalnym, pojawiły się w historii sztucznej inteligencji stosunkowo wcześnie. Na przykład firma AIC, po pięciu latach intensywnych prac, wprowadziła na rynek w 1980 roku **system zapytań Intellect** z dostępem w języku naturalnym. Program ten został przyjęty w 1983 roku przez IBM i udostępniony użytkownikom dużych komputerów. W 1985 roku AIC podpisała podobne porozumienie z firmą DEC.

System Intellect działa obecnie na różnych komputerach, zapewniając komunikację z bazami danych w języku naturalnym. Dla każdego zastosowania buduje się specjalny program pośredniczący. Na przykład, w przypadku dBase II skonstruowano, za pomocą systemu Intellect, program pośredniczący, który dla każdego pytania zadanego przez użytkownika po angielsku generuje odpowiedni program w języku SQL. Wygenerowany program jest przekazywany do dBase II i wykonywany. Wyszukane informacje są przesyłane z powrotem do systemu Intellect, a następnie do użytkownika.

Podstawowymi funkcjami każdego uniwersalnego systemu zapytań w języku naturalnym są: definiowanie słownika dla określonej dziedziny zastosowań oraz automatyczne posługiwanie się nowo wprowadzonym słownictwem na etapie komunikacji z użytkownikiem.

Dostęp głosowy

Jeśli uznać realizację dostępu do programów komputerowych w języku naturalnym za właściwy krok w kierunku ułatwienia komunikacji z użytkownikiem, to następnym krokiem winno być wyeliminowanie klawiatury.

Niektóre urządzenia, takie jak ekrany dotykowe (ang. touch screens) czy też myszy, ułatwiają w pewnym stopniu komunikację, lecz nie mogą całkowicie zastąpić klawiatury. Jeśli użytkownik ma mieć możliwość zadawania różnych pytań, to komunikacja przez wskazywanie wybranych elementów na ekranie jest często nie wystarczająca. Naturalnym sposobem wprowadzenia dodatkowych słów lub zdań jest po prostu wypowiedzenie ich na głos.

Rozwój metod dostępu głosowego następował wolniej niż rozwój metod przetwarzania języka naturalnego. Typowe systemy z dostępem w języku naturalnym używają słowników o zawartości od kilkuset do kilku tysięcy słów, natomiast zasób słów najlepszych systemów z dostępem głosowym nie przekraczał dotychczas 200—300 słów. Ostatnio zaczęły pojawiać się na rynku systemy głosowe, które pokonują tę barierę i umożliwiają budowę praktycznych systemów zapytań z dostępem głosowym.

Na przykład system głosowy **KVS** (Kurzweil Voice System 3000) firmy Kurzweil AI Inc. jest w stanie rozpoznać 1000 słów lub zwrotów z dokład-

nością 97%, porównując wypowiedziane dźwięki z 3000 tzw. symboli głosowych, które zostały uprzednio wyodrębnione z mowy danego użytkownika, przeanalizowane i zapamiętane. Słowa przekazywane do komputera należy wymawiać osobno z przerwami o długości ułamka sekundy. System rozpoznaje wymówione słowo w ciągu 200 ms, co w praktyce nie narzuca żadnego ograniczenia na szybkość mówienia.

System głosowy KVS dołącza się do mikrokomputera typu IBM PC/XT lub AT za pomocą łącza szeregowego RS-232. Możliwe jest zastosowanie dowolnego urządzenia wejściowego, np. słuchawek z mikrofonem, słuchawek telefonicznych lub bezprzewodowych mikrofonów radiowych.

Konstruowanie nowego słownika jest nieskomplikowane. Najpierw użytkownik odczytuje głośno standardowy tekst, co pozwala systemowi określić matematyczną charakterystykę głosu. Następnie użytkownik „uczy" maszynę, wymawiając poszczególne słowa, które mają utworzyć słownik z danej dziedziny. Dla każdego słowa system konstruuje podstawowy symbol głosowy oraz trzy dodatkowe symbole głosowe, co zwykle wystarcza do pełnego zdefiniowania każdego słowa. Dla każdego symbolu głosowego użytkownik musi podać jego reprezentację znakową w postaci napisu ASCII (w większości wypadków jest to zapis ortograficzny danego słowa).

Każdemu wypowiedzianemu słowu system przyporządkowuje najbliższy mu akustycznie symbol głosowy, a następnie wyszukuje przypisaną mu reprezentację znakową. Reprezentacja ta jest przekazywana następnie do mikrokomputera i traktowana tak, jak gdyby był to ciąg znaków wprowadzonych bezpośrednio z klawiatury. Po wykonaniu niewielkich modyfikacji klawiatura może komunikować się również bezpośrednio z systemem KVS. Ciągi znaków ujęte w nawiasy kątowe są kierowane wówczas do systemu głosowego, a nie do mikrokomputera.

Integracja dostępu w języku naturalnym i dostępu głosowego

System głosowy KVS może w praktyce — dzięki dużej objętości budowanych słowników — współpracować z systemami zapytań w języku naturalnym, opracowanymi dla dużych komputerów, np. z systemem zapytań Intellect. Mikrokomputery typu IBM PC, wyposażone w płytę IRMA 3278, są coraz powszechniej używane jako terminale dużych komputerów. Daje to praktyczną możliwość realizacji dostępu głosowego do systemów wyszukiwania informacji.

W pierwszej fazie realizacji definiuje się zasób słów niezbędnych dla danego zastosowania. Następnie buduje się napisy, które będą interpretowane jako polecenia dla systemów operacyjnych mikrokomputera i komputera głównego. W ten sposób złożoność systemu liczącego może być ukryta przed użytkownikiem. Przykładowo, posługiwanie się płytą IRMA może być istotnie uproszczone, jeśli wymówione słowo „logon" zostanie zamienione na ciąg znaków zawierających polecenia wywołania programu emulacji terminala, połączenia z komputerem głównym i przekazania hasła użytkownika.

Napisy, które system KVS zamierza wysłać do komputera głównego są zawsze wyświetlane na ekranie mikro-

Systemy ekspertowe w procesach technologicznych

Wykorzystanie systemów ekspertowych nie ogranicza się do typowych zastosowań diagnostycznych. Obecnie prowadzi się również obiecujące próby ich użycia do zadań sterowania procesami technologicznymi. Wprowadza się je tam ze względu na potrzebę:

- uniknięcia psychicznego przeciążenia operatora, gdy w krytycznej sytuacji liczba otrzymywanych komunikatów o niewłaściwym działaniu przekracza zdolności interpretacji ludzkiej,
- optymalizacji wykorzystania zasobów procesu,
- szybkiej lokalizacji awarii i usunięcia jej przyczyn w sposób zapewniający jak najkrótszą przerwę w działaniu systemu.

Aby osiągnąć te cele należy zapewnić realizację różnego rodzaju zadań, takich jak interpretacja danych umożliwiająca ocenę sytuacji, diagnozowanie niewłaściwego działania systemu, dokładne przewidywanie długo- i krótkoterminowych wymagań związanych z przebiegiem procesu. Do problemów tych stosowano od wielu lat klasyczne podejście algorytmiczne, jednak z niepełnym powodzeniem. Główną trudnością przy takim podejściu była konieczność budowy numerycznego modelu sterowanego procesu. Na ogół, w wypadku złożonych procesów technologicznych, nie dysponuje się pełnymi modelami tego rodzaju. Techniki stosowane w systemach ekspertowych umożliwiają operowanie informacją niepełną, niesprecyzowaną lub dającą się opisać jedynie jakościowo.

W większości zaawansowanych systemów sterowania w czasie rzeczywistym jednym z najważniejszych i najtrudniejszych zadań jest zebranie danych pomiarowych, powiązanie ich i zinterpretowanie. Przetworzona informacja pozwala operatorom szybko zrozumieć aktualną sytuację i właściwie na nią zareagować. Chociaż we współczesnych systemach sterowania stosuje się dość szeroko interpretację danych polegającą m.in. na analizie tendencji i prognozowaniu, wydaje się, że są to już granice możliwości wykorzystywanych obecnie technik.

W systemach konwencjonalnych brak jest przede wszystkim informacji o związkach przyczynowych zachodzących w zbiorze danych pomiarowych. Głównym zadaniem takich systemów jest zebranie pełnej informacji, natomiast w systemach ekspertowych nacisk jest położony raczej na całkowite zrozumienie konkretnej sytuacji. Systemy ekspertowe nie wymagają w danej chwili dostępu do pełnej informacji, muszą natomiast istnieć środki do uzyskania jej wtedy, gdy zajdzie tego potrzeba.

W typowym systemie sterowania można zastosować następujący podział zadań. Urządzenie sterujące przeprowadza obserwację i wyświetlanie danych oraz jest odpowiedzialne za sterowanie. Powiązany z nim system ekspertowy odpowiada za cały proces rozwiązywania problemu. Zajmuje się on oceną sytuacji, przetworzeniem danych i diagnostyką niewłaściwego działania. System ten nie jest uaktywniany jedynie w sytuacjach awaryjnych, lecz działa w sposób ciągły. Prowadzony przez niego nadzór stanowi stałą pomoc dla operatora.

Do tej pory większość systemów ekspertowych stosowano przede wszystkim do diagnozowania. Systemy ekspertowe przeznaczone do sterowania mają inne cechy i charakteryzują się:

- dużą złożonością spowodowaną dużą liczbę składowych,
- dużą liczbą operacji wejścia-wyjścia,
- dynamiką sytuacji, która wymaga uwzględnienia w mechanizmach wnioskowania zależności czasowych, tj. opóźnień między wystąpieniem zdarzeń a ich zarejestrowaniem, przekazaniem poleceń do wykonania a reakcjami na nie, zarejestrowaniem danej sytuacji a końcową diagnozą,
- koniecznością wbudowania mechanizmów uczenia się w celu uwzględnienia zmian w systemie, zapewnienia pracy z początkowo niepełną wiedzą, a także uwzględnienia ewentualności, że wiedza, którą się dysponuje we wczesnych fazach budowy systemu, może być fałszywa.

Szczególnie trudnym do rozwiązania problemem występującym w zastosowaniach związanych z procesami tech-

komputera, co pozwala użytkownikowi upewnić się, czy jego ustne polecenia zostały właściwie zinterpretowane. System daje możliwość przedefiniowania i uaktualnienia słownika. Ciągi znaków mogą być modyfikowane za pomocą klawiatury bądź też głosowo (oczywiście po odpowiednim rozszerzeniu słownika). System Intellect jest wyposażony w mechanizm wykrywania błędów ortograficznych, co umożliwia w pewnych wypadkach poprawianie słów niewłaściwie rozpoznanych lub niepoprawnie użytych przez mówiącego. Na przykład, Intellect może sam usuwać lub dodawać typowe końcówki (takie jak **-ing** lub **-s** w języku angielskim), co oznacza, że nie jest konieczna stuprocentowa dokładność systemu głosowego.

System umożliwia proste i szybkie konstruowanie sprzęgów głosowych. Przykładowo, sprzęg głosowy do obsługi systemu kadrowego o zasobie 700 słów zbudowano w ciągu trzech dni. W tym celu zastosowano mikrokomputer IBM PC/AT wyposażony w system KVS, dysk stały 20 MB, płytę IRMA 3278 oraz niezbędne oprogramowanie. Płytę IRMA dołączono do sterownika terminala IRMA 3274, a następnie — do komputera głównego IBM 3083 z zainstalowanym systemem Intellect.

Jeden system głosowy może obsłużyć kilku użytkowników, lecz nie jednocześnie. Każdy z nich przechowuje swój słownik na dysku elastycznym i wczytuje go do systemu przed rozpoczęciem pracy. Dane niezbędne do zdefiniowania wymienionego słownika o zawartości 700 słów zajmują około 100 KB pamięci.

Aktualnie słownik systemu KVS musi być konstruowany na mikrokomputerze z wykorzystaniem programów dostarczonych przez firmę Kurzweil AI Inc. Oznacza to, że dla każdego zastosowania należy powtórzyć na mikrokomputerze pracę, która została uprzednio wykonana na etapie budowy słownika dla systemu Intellect pracującego na komputerze głównym. Każda zmiana wprowadzona do jednego słownika wymaga więc odpowiedniej zmiany w drugim. Przyszłe wersje systemów Intellect i KVS będą wyposażone w mechanizm automatycznego przekazywania słowników między systemami.

Planuje się również zwiększenie możliwości samego systemu KVS. System ma architekturę opartą na magistrali, do której dołączono trzy płyty. Płyta procesora akustycznego filtruje sygnał głosowy i przekazuje go do płyty głównej, zawierającej procesor mowy, który nadzoruje pracę całego systemu. Trzecia płyta przechowuje w pamięci symbole głosowe i porównuje je z wypowiedziami użytkownika, przekazując do płyty głównej listę słów wraz z prawdopodobieństwem wykonania poprawnej ich identyfikacji. Aby powiększyć zasób rozpoznawanych słów, należy zainstalować większą liczbę płyt dopasowujących wypowiadane słowa do zarejestrowanych symboli głosowych (płyta główna może sterować pracą 15 płyt tego typu). Każda płyta zwiększa zasób słów o 1000 (3000 symboli głosowych). Wkrótce ma pojawić się na rynku model, który będzie w stanie identyfikować z tą samą szybkością 10 000 słów.

M. MACHURA
na podst. „Systems International", styczeń 1987

---

nologicznymi są sytuacje, na które trzeba reagować natychmiast. System sterujący musi nie tylko podejmować poprawne decyzje, ale też działać wystarczająco szybko, aby nadążyć za ciągle zmieniającymi się parametrami nadzorowanego procesu.

W zasadzie do tej pory w dziedzinie sztucznej inteligencji nie zajmowano się problemami czasu rzeczywistego ani od strony oprogramowania, ani od strony sprzętu. Specjalizowane komputery oparte na Lispie nie mogą obecnie obsługiwać przerwań nadchodzących z zewnątrz, w taki sposób, w jaki czynią to komputery stosowane do sterowania procesami technologicznymi, a problem efektywnego gospodarowania pamięcią (ang. garbage collection) w Lispie, który prędzej czy później może doprowadzić do przerwania każdego programu, nie został dotąd rozwiązany.

Problemami zastosowania systemów ekspertowych w procesach technologicznych zajęto się m.in. w RFN, w ramach ogólnokrajowego programu badawczego. Przyjęte rozwiązanie polega na wprowadzeniu inteligentnego sprzęgu między systemem ekspertowym a modułem sterowania procesem technologicznym. Zadaniem tego sprzęgu jest ocena sytuacji i ustalenie na tej podstawie kolejności zadań przeznaczonych do wykonania przez system ekspertowy. Przy takim podejściu, wymagania dotyczące czasu są oddzielone od systemu ekspertowego, a odpowiedzialnością za ich realizację jest obciążony inteligentny sprzęg (w jego ramach sprostanie wymogom czasu rzeczywistego jest już łatwiejsze).

Powyższe założenie powoduje, że struktura proponowanego systemu ekspertowego jest inna niż w tradycyjnych rozwiązaniach z zakresu sztucznej inteligencji. Ogólna koncepcja systemów ekspertowych jest oparta na współdziałaniu z użytkownikiem, który dostarcza informacji niezbędnych w procesie wnioskowania. W tym wypadku system ma do czynienia z informacjami stosunkowo „wysokiego poziomu", posługując się interpretacją badanej sytuacji i ocenami użytkownika. W zastosowaniach związanych z procesami technologicznymi takie podejście traci sens, ponieważ operator nie jest w stanie wprowadzać interakcyjnie wszystkich istotnych dla procesu danych. Ilość danych jest zbyt wielka, a ponieważ wszystkie dane są uwarunkowane czasowo, ręczne ich wprowadzanie spowodowałoby niedopuszczalne opóźnienia. Z tego względu system ekspertowy musi komunikować się bezpośrednio z urządzeniem sterującym, operując danymi „niskiego poziomu" i dokonując oceny sytuacji zależnej w dużej mierze od czasu.

Koncepcja taka nie jest prostym rozszerzeniem metod stosowanych obecnie w systemach ekspertowych, lecz wymaga poważnego wysiłku badawczego, który doprowadziłby do opracowania nowych technologii. Ich pomyślna realizacja będzie mieć wpływ na rozwój złożonych procesów przemysłowych — nie tylko dzięki większej niezawodności sterowania czy też większej wydajności produkcji, ale także dzięki temu, że ułatwi ona stosowanie nowych zasad wykorzystania komputerów w przemyśle.

Wśród możliwych zastosowań systemów ekspertowych w procesach sterowania istotną rolę odgrywają zastosowania związane z dystrybucją energii elektrycznej, gdzie chodzi o nadzorowanie i regulację złożonych i dużych systemów. Jednym z najważniejszych zadań wykonywanych przez personel sterowni (oprócz zadań eksploatacyjnych czy reagowania na niewłaściwe działanie systemu) jest zarządzanie wielkością obciążenia sieci. Ma ono bezpośredni wpływ na ekonomiczną stronę rozdziału energii elektrycznej.

Zarządzanie wielkością obciążenia oznacza m.in. konieczność nadzorowania i dobrego zrozumienia warunków wystąpienia zapotrzebowania na energię, a także — konieczność przewidywania zapotrzebowania w przyszłości. Przewidywania te, których obecnie dokonują operatorzy na podstawie własnych doświadczeń i intuicji, opierają się na dwóch przesłankach:

- porównaniu aktualnej sytuacji z podobnymi sytuacjami, które wystąpiły w przeszłości,
- przewidywaniu zapotrzebowania na podstawie aktualnych czynników zewnętrznych, np. warunków pogodowych oraz dobrego zrozumienia zachowań odbiorców energii elektrycznej. Ten ostatni aspekt musi być oparty na modelu zachowania się odbiorców, co tylko częściowo można opisać ilościowo.

Oprac. PIOTR STRZAŁKOWSKI
na podst. „Systems International", listopad 1986

Pamiętaj o zaprenumerowaniu
INFORMATYKI

CeBIT '87

Decyzja wydzielenia od 1986 r. z programu Międzynarodowych Targów Hanowerskich ekspozycji informatycznej CeBIT, która od ponad 15 lat stanowiła ich część składową, okazała się wyjątkowo trafna. Szybki rozwój informatyki w latach siedemdziesiątych i na początku osiemdziesiątych wywoływał bowiem w Hanowerze systematyczny wzrost zapotrzebowania na powierzchnię wystawienniczą ze strony coraz liczniejszych dostawców sprzętu, oprogramowania i usług. Z chwilą pojawienia się rewolucji mikrokomputerowej zapotrzebowania tego nie mogła zaspokoić nawet specjalnie wybudowana dla ekspozycji CeBIT największa na świecie hala wystawowa nr 1 o powierzchni ok. 120 000 m². Ekspansja elektronicznego przetwarzania danych we wszystkich dziedzinach gospodarki i życia społecznego spowodowała, że w 1985 r. mimo objęcia więcej niż połowy powierzchni hal targowych, CeBIT nie mógł już przyjąć znacznej liczby zgłoszeń potencjalnych wystawców. Z drugiej strony dominacja ekspozycji CeBIT spowodowała odczuwalne zepchnięcie wszystkich pozostałych dziedzin przemysłu na drugi plan, łamiąc tradycyjną koncepcję uniwersalnych targów przemysłowych. Jednocześnie wspomniane przymusowe zacieśnianie ekspozycji informatycznej nie pozwalało na konieczne już — ze względu na jej rozmiary oraz potrzeby zwiedzających — grupowanie tematyczne.

Po olbrzymim sukcesie CeBIT '86 jako oddzielnej imprezy (ponad 330 000 zwiedzających) można było stwierdzić, że pod względem liczby zwiedzających, wystawców i powierzchni wystawienniczej wysunęła się ona zdecydowanie na pierwsze miejsce na świecie, pozostawiając daleko w tyle swój pierwowzór — słynną paryską wystawę SICOB oraz, traktowane łącznie, dwie doroczne amerykańskie wystawy informatyczne COMDEX (wiosenna i jesienna). Oprócz gigantycznej ekspozycji, CeBIT stale towarzyszy kilkanaście imprez typu konferencyjno-seminaryjnego, wprawdzie mniejszej rangi niż związana z wystawą SICOB międzynarodowa konferencja CONVENTION INFORMATIQUE, ale pozwalających rozszerzać wiedzę teoretyczną na temat nowych osiągnięć i rozwiązań oraz wymieniać poglądy z wybitnymi specjalistami w spotkaniach dyskusyjnych. Należy spodziewać się, że światowa pozycja CeBIT oraz doskonałe warunki lokalowe pomieszczeń konferencyjnych w obiektach targowych zachęcą w przyszłości do organizowania w Hanowerze — wzorem SICOB-u — wielkich międzynarodowych konferencji informatycznych.

CeBIT '87 trwał osiem dni (4—11 marca) i mimo bardzo obfitych opadów śniegu w przeddzień otwarcia padł nowy rekord frekwencji: łącznie ponad 400 tys. zwiedzających (SICOB ok. 175 tys.). Mimo zwiększenia powierzchni wystawowej do ok. 300 tys. m² zdołano uwzględnić zgłoszenia „zaledwie" 2200 wystawców (SICOB ok. 800) z 35 krajów. Kolejny raz nie można było przyjąć wszystkich zgłoszeń nowych firm, a wielu dotychczasowym wystawcom trzeba było odmówić zwiększenia powierzchni stoisk. Wg informacji organizatorów, szczególnie duże zapotrzebowanie na dodatkową powierzchnię zgłosili wystawcy japońscy. Pełne uwzględnienie ich potrzeb oznaczałoby opanowanie przez Japończyków prawie całej powierzchni wystawowej (bez hali nr 1, gdzie wszystkie stoiska stanowią własność firm inwestujących przed laty w budowę tego obiektu), a więc praktycznie wyeliminowanie z ekspozycji znacznej części konkurentów. Przekonywującym dowodem determinacji i nieliczenia się z kosztami były stoiska czołowych producentów japońskich, które wyróżniały się swą przestrzennością, a zarazem kontrastowały z ciasnotą w stoiskach wszystkich praktycznie znanych firm zachodnioeuropejskich i amerykańskich, nie wyłączając nie najbiedniejszej przecież firmy IBM.

Po raz pierwszy uwidoczniła się w sposób odczuwalny obecność firm z rozwijających się krajów Dalekiego Wschodu (Tajwan, Korea Południowa, Singapur, Hongkong, Malezja). Mimo atrakcyjnych ofert były one, w przeciwieństwie do firm japońskich, bardzo słabo przygotowane do rzeczowej obsługi zainteresowanych (brak lub zły poziom drukowanych materiałów informacyjnych, słabe przygotowanie merytoryczne i językowe personelu stoisk), w czym przypominały nieco stoiska krajów RWPG na wielu imprezach międzynarodowych. Jeśli chodzi o ostatnio wymienione kraje, to na CeBIT '87 można było stwierdzić zauważalny wzrost ich aktywności (zwiększenie rozmiarów i atrakcyjności ekspozycji oraz liczby wystawców). Na czoło wysunęły się tu Węgry (6 wystawców), Czechosłowacja (3 wystawców) oraz NRD i Bułgaria (po 2 wystawców). Polskę reprezentował tradycyjnie tylko Metronex i to ze stosunkowo jeszcze skromną ekspozycją (drukarki mozaikowe), ale fakt ten może już cieszyć i stwarzać nadzieje na przełom godny tradycji tego przemysłu.

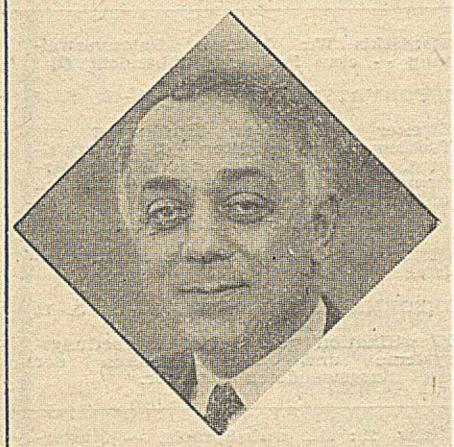
Wspomniane gigantyczne rozmiary ekspozycji praktycznie uniemożliwiają bardziej szczegółowe zapoznanie się z całością ekspozycji, nawet gdyby teoretycznie przeznaczyć na to cały czas trwania wystawy (72 godziny). Stąd koniecznością dla przeważającej części zwiedzających, zwłaszcza specjalistów, staje się rygorystyczne skoncentrowanie zainteresowań wyłącznie na wybranych obszarach tematycznych. Istotną pomocą jest konsekwentnie realizowany przez organizatorów tematyczny rozdział stoisk na poszczególne hale lub grupy hal (np. hala nr 2 — systemy bankowe i ubezpieczeniowe, nr 3 — oprogramowanie i doradztwo, nr 4 — urządzenia peryferyjne, nr 5 — zastosowanie mikrokomputerów, nr 13 16 i 17 — telekomunikacja, nr 18 — systemy CAD/CAM/CIM). W niepełnym jeszcze stopniu udało się to zrealizować w hali nr 1, na której silnie ciąży jej pierwotna organizacja ekspozycji, wynikająca ze wspomnianego już przywiązania stoisk do inwestorów tego obiektu. Oficjalna, mało precyzyjna tematyka tej hali („Büro- und Informationssysteme" — „Systemy biurowe i informacyjne"), wystarczająca, być może, na początku lat siedemdziesiątych, jest nadal utrzymywana i jakby usprawiedliwia tu zróżnicowany charakter ekspozycji. W hali tej dominują prestiżowe, chociaż dziś już zbyt szczupłe stoiska wszystkich wielkich producentów sprzętu komputerowego. Są one wtłoczone w setki stoisk małych producentów i dystrybutorów nie tylko sprzętu i oprogramowania, ale również materiałów i wyposażenia pomocniczego oraz najróżniejszych instytucji tworzących infrastrukturę współczesnej informatyki. Ogromne rozmiary łącznej powierzchni wystawowej oraz wielotysięczna liczba stoisk (wielu wystawców ma ich po kilka) powodują, że odnalezienie, zwłaszcza w hali nr 1, konkretnej firmy lub przedmiotu zainteresowań jest praktycznie niemożliwe bez pomocy przeszło 1000-stronicowego katalogu lub powszechnie dostępnego komputerowego systemu wyszukiwawczego EBi (Elektronische Besucherinformation).

Oprócz wielu zmian na liście wystawców w grupie firm małych, co w wyniku ostrej walki konkrencyjnej i szybkiego rozwoju technologii informatycznej jest zjawiskiem normalnym, na CeBIT '87 ujawniły się zmiany w stabilnej dotąd czołówce wielkich producentów sprzętu komputerowego, co może być początkiem nowych procesów koncentracyjnych. Jedną ze zmian jest zniknięcie dwóch wielkich firm amerykańskich: Sperry-Univac (stale II miejsce po IBM) oraz Burroughs (III—IV miejsce na liście światowej), w miejsce których pojawiło się ich wspólne przedsiębiorstwo Unisys. Przyszłość pokaże, czy zabieg ten oznacza kolejną koncentrację środków do walki z supremacją rynkową IBM, czy też stopniowe wycofywanie się firmy Sperry z rynku komputerowego i skierowanie zainteresowań tego koncernu na inne obszary (m.in. realizacja programu kosmicznego USA).

Zaskakującą zmianą jest również zniknięcie firmy BASF z listy producentów sprzętu kompatybilnego z IBM (duże komputery oraz pamięci masowe). Producenta tego zastąpiło wspólne przedsiębiorstwo koncernów BASF oraz Siemens o nazwie Comparex. Połączenie sił obu gigantów przemysłowych RFN stanowi może silniejsze zagrożenie pozycji IBM, zwłaszcza na rynku europejskim, niż wspomniana fuzja producentów amerykańskich (Siemens w produkcji dużych komputerów opiera się od kilku lat na ścisłej współpracy z japońską firmą Hitachi). Firma BASF, będąca przecież koncernem chemicznym, pozostała jednak na rynku informatycznym jako producent doskonałej jakości magnetycznych nośników informacji oraz materiałów eksploatacyjnych do drukarek laserowych i kopiarek

dokończenie na str. 38

Kto jest kim w IFIP



Jean D. Ichbiah

Jean D. Ichbiah urodził się w 1940 roku w Paryżu. W wieku 20 lat wstąpił do Ecole Polytechnique, jednej z najbardziej obleganych szkół inżynierskich Francji, założonej jeszcze przez Napoleona dla kształcenia urzędników państwowych. Po dwóch latach intensywnych studiów otrzymał stopień inżyniera. Pamięta on profesora Favarda, który na pierwszym wykładzie matematyki powiedział: *Nie jest naszym celem kształcenie inżynierów mających kierować przedsiębiorstwem, które już dobrze działało bez ich udziału.* Ichbiah nie pragnął niczego więcej — wierzył, że inżynier musi widzieć efekty swojej pracy i nie mógłby pogodzić się z faktem, że jest kimś, czyja obecność lub jej brak nie robi żadnej różnicy. Po zakończeniu Ecole Polytechnique przez rok odbywał służbę wojskową, a następnie kształcił się w Ecole des Ponts et Chausses.

W 1965 roku Jean D. Ichbiah otrzymał stypendium na studia w USA nad zastosowaniem informatyki w inżynierii lądowej. Spędził dwa lata na Wydziale Transportu MIT, gdzie projektowano języki użyteczne dla inżynierii lądowej. W tym okresie znajdował się m.in. pod wpływem wykładów Toma Cheathama i Jacka Dennisa, a także pracy Wirtha „On Simple Precedence". Po otrzymaniu tytułu doktora filozofii Jean D. Ichbiah powrócił do Francji, gdzie początkowo pracował w Instytucie Badawczym Transportu, zakładając tam dział informatyki. Praca ta polegała głównie na nabywaniu komputerów i kierowaniu dwudziestoma podwładnymi. Po upływie dziewięciu miesięcy skorzystał z możliwości przeniesienia się do CII.

Wówczas we Francji powoli zaczynano rozumieć znaczenie przemysłu komputerowego. Pierwszym przedsięwzięciem Ichbiaha w CII było opracowanie kompilatora Simuli. W rzeczywistości, dwa pierwsze kompilatory Simuli powstały właśnie we Francji — jeden w firmie CDC, a drugi w CII. Zespół projektowy składał się z 2—3 inżynierów, a pozostałą jego część stanowili coraz to inni studenci. Ponieważ prace trwały trzy lata, Jean D. Ichbiah był zmuszony zastosować pewien rodzaj dyscypliny projektowej, zwykle zalecanej tylko w dużych pracach programistycznych. Niedoświadczeni programiści musieli być szybko przeszkoleni, aby tworzyć programy, które potem mogłyby być pielęgnowane przez innych, niedoświadczonych programistów.

W tym czasie Dijkstra opublikował słynny dziś artykuł o instrukcji skoku „goto". Ichbiah odebrał go bardzo pozytywnie, a jego przesłanie odczytał następująco: *program należy traktować jako obiekt statyczny i niezależny od wykonania.* Ichbiah uznał ten artykuł za początek nowoczesnych prac metodologicznych. Miał on duży wpływ na strukturalizację tworzonego kompilatora Simuli. Inne decyzje były skutkiem implementacji w języku LP70 (język podobny do PL/360). Choć stosowne koncepcje umożliwiały tworzenie czytelnych programów, język miał także pewne niebezpieczne cechy — w pierwszym rzędzie wrażliwe instrukcje wyboru. Analiza błędów tej instrukcji uzmysłowiła Ichbiahowi, że zrozumienie źródeł pewnych grup błędów umożliwiałoby zaprojektowanie lepszej składni. To badanie „błędów charakterystycznych" doprowadziło do bezpieczniejszych form instrukcji wyboru w Pascalu, a potem w Adzie.

Wpływ tych trzech lat sprawił, że Jean Ichbiah uwierzył, iż poważna praca wymaga dostępności lepszych języków programowania. Od czasu stworzenia opłacalnego kompilatora Simuli, posiadł on ogromny kredyt zaufania u swego kierownictwa. Z kredytu tego skorzystał, aby przekonać kierownictwo CII, jak ważne jest uwzględnienie czytelności, niezawodności i pielęgnowalności przy projektowaniu nowego języka. Wkrótce też rozpoczął, wraz z Jeanem Claudem Heliardem i Jeanem Paulem Rissenem, pracę nad językiem LIS. Potem dołączył do nich jeszcze Patric Cousot. Pierwszą wersję języka ukończono w 1972 roku, a główne poprawki wprowadzono w roku 1974. Podczas tej pracy Jean D. Ichbiah zaczął formułować idee, które miały czekać wiele lat na szeroką akceptację. Koncepcja pakietu Ady po raz pierwszy pojawiła się właśnie w „partycjach" LIS-a, gdzie oddzielono specyfikację od ciała. Praca ta poprzedzała opracowanie języków Alphard i CLU. Na seminarium SIGPLAN/SIGOPS, w Savannah, w 1972 roku Ichbiah zaprezentował niektóre ze swoich wcześniejszych prac.

Jean D. Ichbiah przez wiele lat był aktywnym członkiem grup projektujących języki profesjonalne. Wraz z Billem Wulfem i Jimem Horningiem był jednym z trzech założycieli Grupy Roboczej IFIP WG 2.4 ds. Języków Programowania Systemowego. Był także członkiem grupy LTPL (ang. Long Term Procedural Lanuguage), działającej w ramach International Purdue Workshop on Industrial Computer Systems. Obie te grupy były doskonałym forum do dyskusji o koncepcjach nowoczesnych języków, w szczególności znalazło się tam kilka osób, które wniosły duży wkład w rozwój Ady. Podczas udział w pracach grupy LTPL Ichbiah poznał Billa Whitakera i cele przedsięwzięcia polegającego na zaprojektowaniu nowego języka wysokiego poziomu, finansowanego przez amerykański Departament Obrony. Po przestudiowaniu założeń przedstawionych w raportach Tinman i Ironman (por. Informatyka, nr 8,1981) Ichbiah stwierdził, że generalnie skłania się do filozofii Ironmana — próbował przekonać grupę LTPL do przyjęcia jego wytycznych.

W pierwszym etapie konkursu Jean D. Ichbiah przedstawił swoją ofertę i został wybrany jednym z czterech projektantów do dalszego współzawodnictwa. Jego pierwszą decyzją było przyjęcie wstępnej definicji języka za dokument przeglądowy. Ten pierwszy dokument objaśniał projekt, nazwany później Green, w sposób ukazujący jego strukturę i estetykę tak, że został przyjęty entuzjastycznie. W drugim etapie konkursu Ichbiah udoskonalił pierwotny dokument i odpowiedział na komentarze do pierwszego etapu. Język, który powstał w tym czasie dowodził, że jego podstawy były ustalone prawidłowo. Green stał się Adą. W ciągu następnych dwóch czy trzech lat Ichbiah skoncentrował się na dopracowywaniu Ady. Recenzenci, którzy analizowali funkcjonalność języka z punktu widzenia zastosowań, wraz z grupą Ichbiaha rozszerzyli propozycje w zakresie architektury Ady. Ichbiah wiedział, że był często zbyt autokratyczny, lecz czuł, że silne przywództwo jest potrzebne do zachowania jednolitości języka. Ada nie miała być językiem zaprojektowanym przez komitet, lecz przez małą grupę z silnym kierownictwem, ale przy pomocy komitetu doradczego.

Po ukończeniu Ady Ichbiah opuścił CII i założył własną firmę — Alsys. Obecnie pracuje nad kompilatorami tego języka na komputery VAX i i IBM PC/AT. Za projekt Ady otrzymał wiele nagród m.in. Francuską Legię Honorową.

Jean D. Ichbiah ma wraz z żoną Marianne troje dzieci. Oprócz Ady włada on francuskim, niemieckim, włoskim, angielskim i hiszpańskim. Uprawia judo (od 12 lat ma brązowy pas) i windsurfing.

Oprac. MARIUSZ KUC

na podst. Ada Letters, Vol. 2. No. 6

Jones C. B.: Specyfikacje a programy (2)

INFORMATYKA 1987, nr 11—12, s. 2

Druga część charakterystyki metody VDM w zastosowaniu do konstruowania oprogramowania, zawierająca omówienie obszernego przykładu oraz różnic w porównaniu do innych metod.

Jones C. B.: Specifications and programs (2)

INFORMATYKA 1987, No. 11—12, p. 2

Second part of characteristics of the VDM method and its application for software constructing, which contains discussion of a vast example and differences in comparison with other methods.

Jones C. B.: Spezifikationen und Programme (2)

INFORMATYKA 1987, Nr. 11—12, S. 2

Zweiter Teil einer Charakteristik von VDM-Methode und ihrer Anwendung zur Softwarekonstruktion, der eine Besprechung von einem umfassenden Beispiel und Unterschiede im Vergleich zu anderen Methoden umfasst.

Staniszkis W.: Zarządzanie rozproszonymi danymi — przegląd problematyki (2)

INFORMATYKA 1987, nr 11—12, s. 7

Druga część wprowadzenia do problematyki zarządzania rozproszonymi danymi, zawierająca omówienie systemów z wieloma bazami danych.

Staniszkis W.: Distributed data management — problems survey (2)

INFORMATYKA 1987, No. 11—12, p. 7

Second part of an introduction to distributed data management problems, which contains discussion on multibase systems.

Staniszkis W.: Verteile Datenverwaltung — eine Problematikübersicht (2)

INFORMATYKA 1987, Nr. 11—12, S. 7

Zweiter Teil einer Einführung zur Problematik verteilter Datenverwaltung, der eine Besprechung von Systemem mit mehreren Datenbanken umfasst.

Bieńkowski K.: Systemy przetwarzania tekstów

INFORMATYKA 1987, nr 11—12, s. 11

Klasyfikacja systemów przetwarzania tekstów oraz charakterystyki stosowanych w nich drukarek.

Bieńkowski K.: Text processing systems

INFORMATYKA 1987, No. 11—12, p. 11

Classification of text processing systems and characteristics of printers applied there.

Bieńkowski K.: Textverarbeitungssysteme

INFORMATYKA 1987, Nr. 11—12, S. 11

Klassifikation der Textverarbeitungssysteme und Charakteristiken der dort ansgewendeten Drucker.

Płoski Z.: Język awk — przetwarzanie wzorców

INFORMATYKA 1987, nr 11—12, s. 16

Charakterystyka języka programowania awk, przeznaczonego do wyszukiwania informacji oraz operowania danymi.

Płoski Z.: The awk language — pattern processing

INFORMATYKA 1987, No. 11—12, p. 16

Characteristics of the awk programming language for information retrieval and data handling.

Płoski Z.: Die awk-Sprache — Musterverarbeitung

INFORMATYKA 1987, Nr. 11—12, S. 16

Eine Charakteristik der awk-Programmiersprache für Informationrecherche und Datenhandhabung.

Kłopotek M.: Maszynowe tłumaczenie tekstów

INFORMATYKA 1987, nr 11—12, s. 20

Historia, stan obecny oraz perspektywy rozwoju systemów komputerowego tłumaczenia tekstów obcojęzycznych.

Kłopotek M.: Machine text translation

INFORMATYKA 1987, No. 11—12, p. 20

History, recent state and development perspectives of machine text translation.

Kłopotek M.: Maschinelle Textübersetzung

INFORMATYKA 1987, Nr. 11—12, S. 20

Geschichte, heutiger Stand und Voraussichten der Entwicklung von Systemen für maschinelle Übersetzung fremdsprachiger Texte.

Kuroczycki T.: Redaktor 2000 — mikrokomputerowy procesor tekstów

INFORMATYKA 1987, nr 11—12, s. 24

Charakterystyka procesora tekstów uwzględniającego specyfikę języka polskiego dla komputera Mazovia 1016 i komputerów kompatybilnych z IBM PC.

Kuroczycki T.: Redaktor 2000 — a microcomputer text processor

INFORMATYKA 1987, No. 11—12, p. 24

Characteristics of the Redaktor 2000 text processor for Mazovia 1016 and IBM PC compatible microcomputers, which allows to accept specific characters of polish language.

Kuroczycki T.: Redaktor 2000 — ein Mikrocomputer-Textprozessor

INFORMATYKA 1987, Nr. 11—12, S. 24

Eine Charakteristik des Redaktor 2000-Mikroprozessors für Mazovia 1016 und IBM PC kompatiblen Mikrocomputer, der die Eigenschaften der polnischen Sprache berücksichtigt.

Płoski Z.: Chiwriter — procesor tekstu (1)

INFORMATYKA 1987, nr 11—12, s. 27

Pierwsza część charakterystyki procesora tekstów dla mikrokomputerów typu IBM PC.

Płoski Z.: Chiwriter — a text processor (1)

INFORMATYKA 1987, No. 11—12, p. 27

First part of the Chiwriter text processor characteristics, elaborated for IBM PC microcomputers.

Płoski Z.: Chiwriter — ein Textprozessor (1)

INFORMATYKA 1987, Nr. 11—12, S. 27

Erster Teil einer Charakteristik von Chiwriter-Textprozessor für IBM PC-Mikrocomputer.

Ossowska S.: Wordstar 3.30 — zasady działania i sposób użytkowania (3)

INFORMATYKA 1987, nr 11—12, s. 30

Trzecia część charakterystyki zasad działania i sposobu użytkowania pakietu Wordstar 3.30, zawierająca omówienie poleceń zawartych w menu dla ekranu i sterowania wydrukiem, a także tzw. poleceń z kropką.

Ossowska S.: Wordstar 3.30 — operating principles and handling (3)

INFORMATYKA 1987, No. 11—12, p. 30

Third part of the Wordstar 3.30 characteristics, which contains discussion of menu commands for display and printing control, as well as so called commands with point.

Ossowska S.: Wordstar 3.30 — Betriebsgrundsätze und Handhabung (3)

INFORMATYKA 1987, Nr. 11—12, S. 30

Dritter Teil einer Charakteristik von Grundsätzen und Handhabung des Wordstar 3.30-Programmpakets, der eine Besprechung von Menubefehlen für Bildschirm und Drucksteuerung, sowie sogenannte Befehle mit Punkt umfasst.

Syfert A., Raznowiecki A.: Struktura systemu operacyjnego PC-DOS (3)

INFORMATYKA 1987, nr 11—12, s. 32

Trzecia część charakterystyki systemu operacyjnego PC-DOS, zawierająca omówienie funkcji znakowego wejścia-wyjścia oraz zarządzania plikami.

Syfert A., Raznowiecki A.: Structure of the PC-DOS operating system (3)

INFORMATYKA 1987, No. 11—12, p. 32

Third part of the PC-DOS operating system characteristics, which contains discussion of functions for character input-output and file management.

Syfert A., Raznowiecki A.: Struktur des PC-DOS-Betriebssystems (3)

INFORMATYKA 1987, Nr. 11—12, S. 32

Dritter Teil einer Charakteristik von PC-DOS-Betriebssystem, der eine Besprechung von Zeichen E/A- und Dateiverwaltungsfunktionen umfasst.

Bielecki J.: Język C — konstruowanie, interpretowanie i wyznaczanie wartości wyrażeń

INFORMATYKA 1987, nr 11—12, s. 37

Zasady oraz praktyczne wskazówki posługiwania się wyrażeniami języka C.

Bielecki J.: C language — constructing, interpreting and value defining of expressions

INFORMATYKA 1987, No. 11—12, p. 37

Principles and practical advices for using expressions of the C programming language.

Bielecki J.: C-Sprache — Konstruierung, Interpretierung und Wertbezeichnung der Ausdrücke

INFORMATYKA 1987, Nr. 11—12, S. 37

Grundsätze und praktische Hinweise für Anwendung der Ausdrücke von der C-Programmierungssprache.

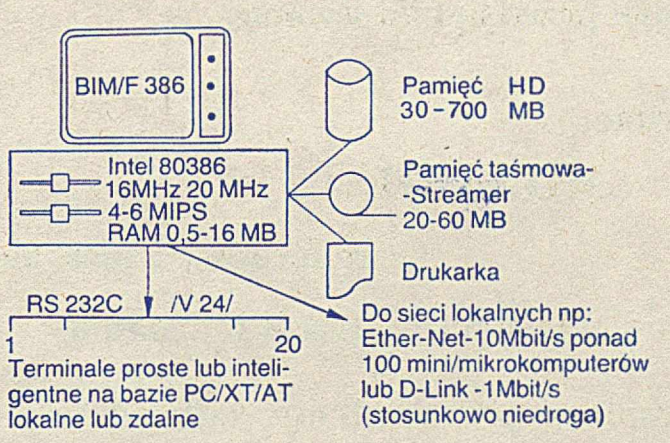
PZ globo®

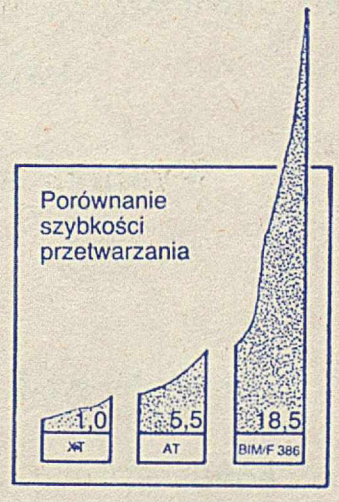
Licencjonowany producent firmy BIM
Koplin 73 200 Choszczno Tel. 7550 Telex 0445413

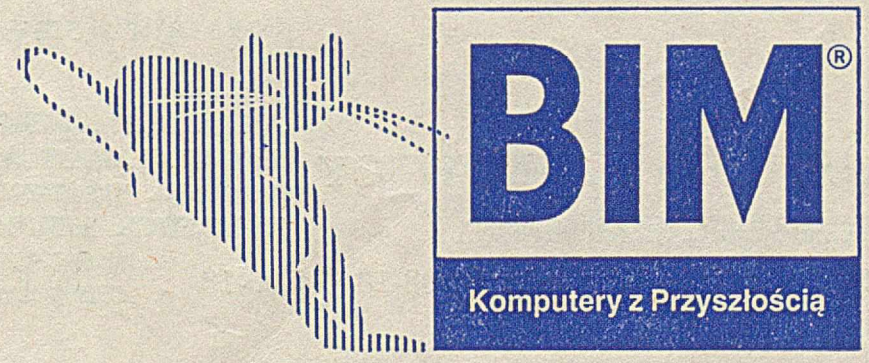
Oferuje nowoczesne, niezawodne, bardzo wydajne:

- Minikomputery 32-bitowe **BIM/F 386** – kompatybilne z **IBM PC/AT**
- Mikrokomputery 16-bitowe **BIM PC/XT/AT** – kompatybilne z **IBM PC/XT/AT**
- Terminale, modemy, koncentratory transmisji danych.
- Systemy użytkowe: **F-K,** materiałowy, kadrowo-płacowy, kosztorysowania.
- Sieci mikrokomputerowe (Ether-Net, D-Link/ oraz systemy wielodostępne na bazie **BIM/F 386, BIM PC/XT/AT** realizowane „pod klucz" – z oprogramowaniem systemowym i użytkowym, terminalami, modemami z instalowaniem i szkoleniem

Przykład sieci opartej o minikomputer 32-bitowy BIM/F 386







BIM jest zastrzeżonym znakiem towarowym firm BIM Technologies AG i PZ Globo
IBM jest zastrzeżonym znakiem towarowym International Business Machines Corporation

EO/614/87

KONIEC TWOICH PROBLEMÓW

Nareszcie wszystkie potrzebne Ci dane dotrą na czas w przejrzystej formie tabel i wykresów.

Pakiet oprogramowania na mikrokomputery 16-bitowe

MEGA — BANK

to nowe MEGA możliwości jakie otwierają się przed Twoim przedsiębiorstwem.

Wyobraź sobie

MILIARD

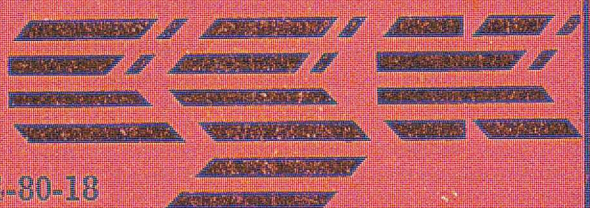
rekordów, które możesz zapełnić według własnych potrzeb i uznania.

Miliard kooperantów, miliard pracowników, miliard produktów — z tym wszystkim nasz MEGA-BANK poradzi sobie bez trudu.

System jest łatwy w obsłudze i opracowany w języku polskim.

Gwarantujemy satysfakcję!

COMPUTER STUDIO KAJKOWSCY

PROFESJONALNE OPROGRAMOWANIE MIKROKOMPUTERÓW

ul. Balladyny 3B, 81-524 Gdynia, tel.: 21-08-18, 24-80-18